Universiteit Leiden

Master Computer Science

[Efficient Event-based Eye Tracking Using A Lightweight Mamba-Based Model]

Name: Xuening Xin
Student ID: S3662012

Date: [13/06/2025]

Specialisation: [Artificial Intelligence]

1st supervisor: [Qinyu Chen]
2nd supervisor: [Chang Gao]

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgements

I would like to express my heartfelt gratitude to my advisor, Dr. Chen Qinyu, and Dr. Chang Gao for your meticulous guidance, patient assistance, and encouragement throughout the entire thesis process. Your expertise and rigorous approach have deeply influenced me and provided a solid foundation for the completion of this thesis.

I would also like to extend my special thanks to doctoral student Lu Guorui for the invaluable advice and guidance you provided throughout the project. Your selfless assistance and patient explanations have helped me grow significantly in my research journey.

Additionally, I am deeply grateful to my lab colleague Zhen Xu for your support and assistance when I faced challenges. Our collaboration has been immensely beneficial to me.

Thank you to my friends, who always offered encouragement and companionship when I was stressed or confused, serving as my strong support system.

Finally, I would like to extend my heartfelt gratitude to my parents for your understanding and support throughout. It is because of you that I have the courage to pursue my dreams.

Abstract

This paper proposes a lightweight, deployable eye tracking model based on event data. Using a cropped custom MobileViTv2 for spatial encoding and the Mamba module for temporal modeling, we propose a lightweight modular architecture that maintains high prediction accuracy while significantly reducing model complexity. Experiments on the 3ET Challenge dataset show that our best model achieved a pixel error of 1.65 using only 265K parameters, representing a 98% reduction in model size compared to the 3ET baseline (from Top-1 ranking team UTCEventGroup in 3ET+ challenge at CVPR 2025). Ablation experiments demonstrated that the Mamba module outperforms traditional GRU and LAF modules in terms of both accuracy and efficiency in the eye tracking task. Additionally, we visualized the pupil center prediction behavior, verifying the model's effective extraction of spatio-temporal features from event data. Finally, this paper discusses several promising directions for future research, including adaptive temporal resolution, online processing mechanisms, multimodal information fusion, and self-supervised learning methods. The above results validate the application potential of lightweight eye-tracking models for edge deployment in event-based visual scenarios.

Contents

1 Introduction

With the wide application of AR/VR/MR technology in education, medical and industrial fields, eye tracking technology has gradually become one of the key means to improve the efficiency of user interaction. By capturing the user's pupil center in real time, the eye tracking system can not only achieve natural interaction based on line of sight, but also support focus detection and attention recognition. In the education field [10], eye tracking can be used to analyze the students' attention patterns during reading, image comprehension, or problem solving, helping teachers understand students' learning behaviors and make personalized teaching plans. In the medical field, companies such as Tobii Dynavox [25] have applied eye tracking to assist people with disabilities in computer operations, enabling users to perform all the functions of a Windows system independently using only their eye movements. In industrial applications, the technology is widely used for tasks such as driver fatigue detection [23] and early warning of hazardous behaviors which effectively improve the safety and responsiveness of the system. With the development of technology, eye tracking is becoming a key component of human-computer interaction, providing strong support for creating a more immersive, intelligent and efficient interaction environment. In our work, we mainly focus on pupil tracking, which is a very important basic subtask in eye tracking. Accurate estimation of the pupil center position is the basis for many downstream tasks (such as gaze point estimation, attention analysis, and eye-tracking-based interaction).

Currently, the mainstream eye tracking devices are mainly frame-based optical eye tracking devices [26]. Frame-based eye trackers usually combine infrared illumination with a high-speed camera to acquire a series of eye images for accurate localization of the pupil, with sampling rates can reach hundreds or even thousands of Hz, which is the most commonly used solution in the scientific research and commercial fields. However, frame-based systems are prone to image blurring and delays in bright light, rapid eye movements or moving environments due to the limitation of fixed frame rate, which makes them limited in eye tracking tasks and thus unable to realize high-precision pupil center capture. In some medical and low-power scenarios, Electro-Oculogram (EOG) devices based on electrophysiological signals [4] have been used to infer the direction of vision by detecting the potential changes caused by eye movements. Although EOG devices have the advantages of low power consumption and not relying on visual images, the limited spatial resolution makes it difficult to meet the demand for eye tracking.

In order to overcome the under-performance problem of frame-based images in high-speed dynamic environments, more and more researches in recent years have

attempted to introduce event vision techniques to enhance eye tracking performance. Event vision sensors have been gradually applied to eye tracking tasks. These sensors have the advantages of microsecond time resolution, low latency, and low power consumption, but their application in eye tracking still faces many unique challenges [14]. First, the raw event streams are inherently sparse, asynchronous and noisy. Unlike traditional frame cameras that acquire a dense pixel grid image with luminance values at each pixel point at fixed intervals, event cameras output asynchronous events only when the luminance of a pixel changes above a certain threshold. This makes it difficult to directly apply traditional convolutional neural networks (CNNs) or recurrent architectures. Furthermore, due to sensor limitations and environmental factors, event data can be severely distorted by noise, especially when rapid eye movements or blink artifacts interfere with the signal stream. These properties make robust representation learning and temporal modeling particularly difficult.

To address these issues, many teams have begun to experiment with more complex architectures. Several eye-tracking methods based on event data have been proposed, especially in tasks such as the 3ET Challenge [7], where some of the leading solutions use a combination of deep CNN backbones, stacked GRUs or attention modules, and post-processing to achieve high accuracy (e.g., pixel error 1.14). However, these architectures tend to have more than 12.89M parameters, high computational effort, slow inference, and high training cost, making them difficult to deploy in resource-constrained environments such as edge platforms and AR/VR devices.

Current studies mainly focus on two aspects: first, how to fully exploit the advantages of event data to improve the prediction accuracy of the model; second, how to construct a lighter and more efficient network structure under the premise of considerable accuracy. However, there is a lack of research on the impact of module (e.g., time-series modeling module, number of layers of spatial feature extraction) on the network structure in terms of performance, parameter size. In particular, there is still no clear answer to whether deep and complex architectures are important for high-precision predictions. This study is driven by a central question: Can high-precision eye tracking task be achieved in event-based vision without relying on deep and complex architectures, making models suitable for edge deployment?

To investigate the trade-off between modeling complexity and prediction accuracy, we would like to explore the combination of simpler, more efficient, and more deployable architectures while maintaining the prediction accuracy. To this end, we designed a modular joint spatial-temporal modeling framework based on MobileViTV2, and compared different architectural variants, examining the replacement

of temporal modeling modules (GRU, LAF, and autoregressive Mamba), and the adjustment of the spatial encoder depth by either retaining deeper layers or applying appropriate pruning (e.g., removing layer 4). Ultimately, we obtained a small model with only 265K parameters but still maintaining high accuracy (Pixel Error 1.65) on the 3ET+ dataset, verifying the feasibility of the lightweight structure.

In summary, our work explores the modular combination approach in the joint spatial-temporal modeling architecture, and proposes a lightweight eye-tracking model that can be efficiently deployed on embedded platforms, which provides a practical paradigm for model compression and structural optimization in event vision tasks.

2 Related Work

2.1 Classical Eye Tracking under Frame-based Vision

Traditional eye tracking methods are mainly based on frame-based image inputs. These image sequences are usually captured at a fixed frame rate by RGB or infrared (IR) cameras. Early methods mostly used traditional image processing techniques such as edge detection, threshold segmentation, and ellipse fitting to estimate the pupil center. For example, the method proposed by Swirski et al. [34] has been widely adopted and is considered as one of the classical schemes. This method is mainly based on the convolution of the eye image with a center-surround structure, followed by K-means clustering segmentation of the corresponding maximum position, further extraction of the boundary using Canny edge detection, and finally ellipse fitting using the RANSAC algorithm [11], to achieve the accurate localization of the eye boundaries and centers. The ExCuSe algorithm [12] uses a multi-stage image processing method to detect the eye center, including Canny edge detection [6], ellipse fitting, coarse localization, and fine correction to make it more robust. This method can gradually eliminate straight line interference and retain local features to accomplish more refined localization, which is more applicable to real-world scenarios. In addition, PuRe [13] is also a robust eye detection algorithm specially designed for real-world environments, combining edge segment selection, conditional segment combination strategy, and confidence assessment mechanism, which can significantly improve detection accuracy and stability. It performs eye localization through structured image analysis pipeline, and have the ability to recognize images that do not contain eyes.

However, these methods are difficult to integrate with modern deep learning frameworks, lack end-to-end modeling capabilities, and are not well-suited for han-

dling high-speed dynamics due to their reliance on frame-based image processing. Although traditional methods are reliable under certain conditions, they tend to be highly rely on image quality and environment which make it difficult to adapt to the needs of real-time, complex or mobile applications. To address these issues, researchers have begun to introduce deep neural networks to learn robust representations for pupil, improving the accuracy and generalization ability of the model.

2.2 Deep Learning for Eye Tracking

In recent years, with the wide application of deep learning techniques, researchers have begun to try to introduce them into eye tracking tasks to achieve stronger representation learning and better robustness. For example, Wei-Liang Ou et al. proposed a wearable eye tracking method based on YOLOv3-tiny network [29], which is specifically designed for near-eye images under visible light conditions, replacing the traditional ellipse fitting method. Detection of the eye region is achieved by training a lightweight YOLOv3-tiny model, which is then combined with a calibration process to achieve pupil center prediction. Lee et al., on the other hand, propose a deep network that combines Self-Attention with a mutual information learning strategy to achieve higher accuracy in eye center detection [21].

In addition, related studies have also explored multi-task structures to jointly model eye detection and pupil center estimation. For example, GazeNet [38] introduces a joint encoding structure for binocular images and head pose, which exhibits significant robustness in cross-dataset scenarios and provides high-quality inputs for pupil center in complex pupil center estimation systems. However, these methods generally rely on image sequences acquired at a fixed frame rate and still have limitations under rapid eye movements, occlusion, or extreme lighting conditions. On the other hand, traditional image sensors suffer from temporal redundancy and limited dynamic range, which limit their further improvement in real-time and robustness. For this reason, more and more research is focusing on the potential of new visual sensors, such as event cameras, for pupil center tracking, with a view to solving the performance bottleneck of traditional frame-based methods in dynamic scenes.

2.3 Event-based Eye Tracking

Currently, event cameras are widely used in a variety of scenarios such as target tracking [9, 17], object/gesture recognition [22, 28, 1], optical flow estimation [39, 5], simultaneous localization and mapping (SLAM) [20, 31, 36], and so on. For example, Bharath et al. [30] proposed the e-TLD (Extended Tracking-Learning-

Detection) framework, which integrates an event-based camera to achieve robust long-term target tracking under cluttered environments and 6-DOF camera motion, demonstrating the unique advantages of event-based cameras for real-time tracking scenes. In the field of optical flow estimation in event vision, Bardow et al. [3] proposed an innovative method to simultaneously estimate the scene luminance map and the pixel-level motion field while relying solely on event flow. With the development of event vision, more and more research is focusing on its application in the field of eye tracking (pupil center Tracking). Unlike traditional frame images, event data is characterized by microsecond time resolution, low latency, and high dynamic range [14], and its kilohertz-level temporal resolution and ability to capture fine details of consecutive eye movements make it an ideal sensor for realizing ultra-high-precision eye tracking [14, 2]. The main event representation methods currently available include:

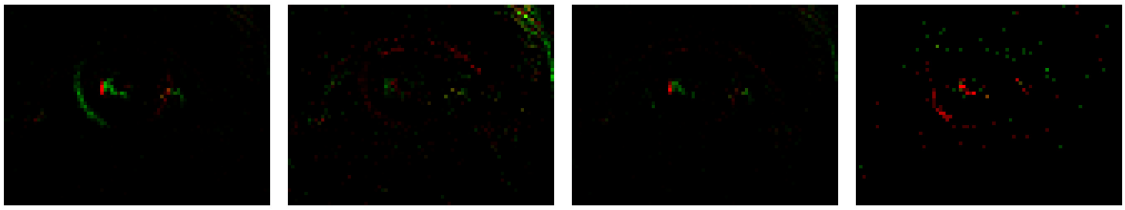


Figure 1: Event frame images from the 3ET dataset [7]. Each frame is generated cumulatively for events through a 10 ms time window. Red pixels in the figure indicate ON events (positive polarity) and green pixels indicate OFF events (negative polarity). The spatial distribution of events depicts the main contours of the ocular structures, especially the pupil region.

- **Time Slicing with Polarity Map**: Event data is binned within a fixed time window (e.g., 10 ms) and accumulated into RGB channels by positive and negative polarity to form pseudo-image inputs, making them suitable for traditional CNN architectures. As shown in Figure 1, this representation effectively converts sparse event streams into dense frame-like inputs.
- **Voxel Grid**: Events are projected into a voxelized 3D tensor to capture the joint distribution over time, space, and polarity [15].
- **Time Surface**: A time map encodes the timestamp of the most recent event at each pixel to model local temporal dynamics [33].

While voxel grids provide higher spatio-temporal richness and temporal ordering, time surfaces are simpler and lighter but more sensitive to sparse or noisy events. However, regardless of the method used to construct the event frame, there are two

common characteristics: asynchrony and sparsity, which make time modeling a critical component. Existing mainstream temporal modeling methods mainly include:

- **Recurrent Networks** (e.g., Bi-GRU), which model temporal dependencies across sequences.
- **Attention Mechanisms** (e.g., BRAT), which enhance long-distance dependency modeling.
- **State Space Models** (e.g., Mamba), which offer efficient global sequence modeling.

Among the existing studies, Mambaeye [37] is a representative method with leading performance in eye tracking tasks under event vision. The method proposes a bi-directional selective temporal modeling framework consisting of a CNN-extracted spatial encoder, a bi-directional GRU, and a linear time-varying state-space module (LTV-SSM). Mambaeye performs well on the ThreeET-plus benchmark dataset, but has a model parameter count of 8.59M with a computational overhead of 2.61T FLOPs. The UTCEventGroup team proposed the BRAT architecture in the 3ET Challenge [7]. As shown in (Figure 2 and Figure 3), which uses a convolutional encoder to extract spatial features and combines Bi-GRU with bidirectional relative positional attention to model temporal dynamics . This approach achieves a maximum accuracy of 1.14 pixel error [6], but again contains multiple complex modules with a parameter count of 12.89M, hindering deployment on edge devices. While these eye-tracking methods have made significant progress in terms of accuracy, their schemes often rely on stacked depth structures and complex attention mechanisms that consume large amounts of computational resources, making it difficult to meet the requirements of low-power, low-latency deployments such as AR/VR, which has prompted researchers to try to find lightweight deployable network architectures.

2.4 Lightweight and Deployable Architectures

To address the problem of difficult model deployment, recent research has begun to explore lightweight and efficient architectures. These approaches aim to strike a better balance between accuracy and deployability by simplifying spatial and temporal encoders, trimming redundant layers, or adopting efficient model designs such as linear state-space models like MobileViT or Mamba. For example, Iddrisu et al. proposed a YOLOv8-n based detection framework for the eye tracking task, which achieves a good balance between very low parameter count and high real-time inference [19] through a modular feature extraction structure with only 3M parameters

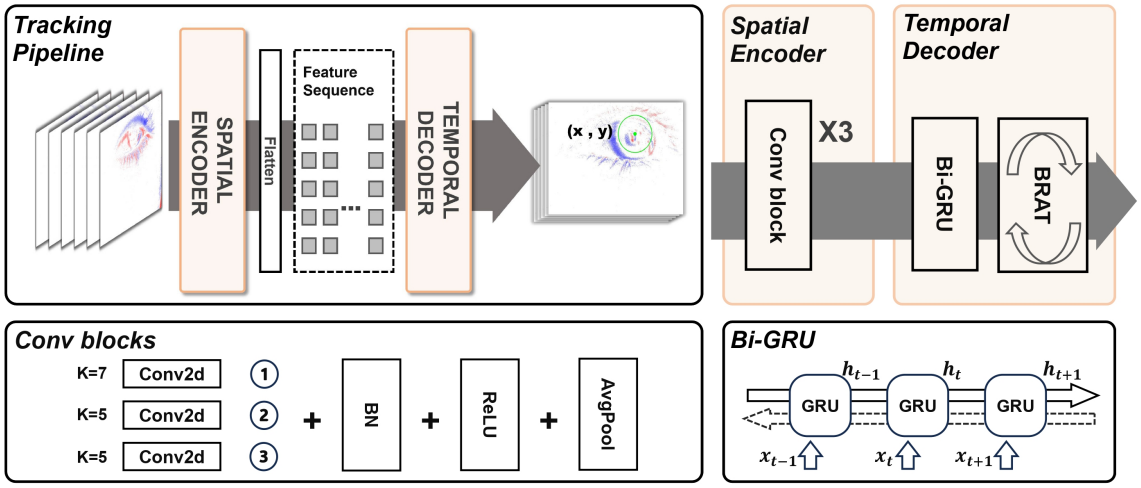


Figure 2: BRAT network by Team USTCEventGroup [7]

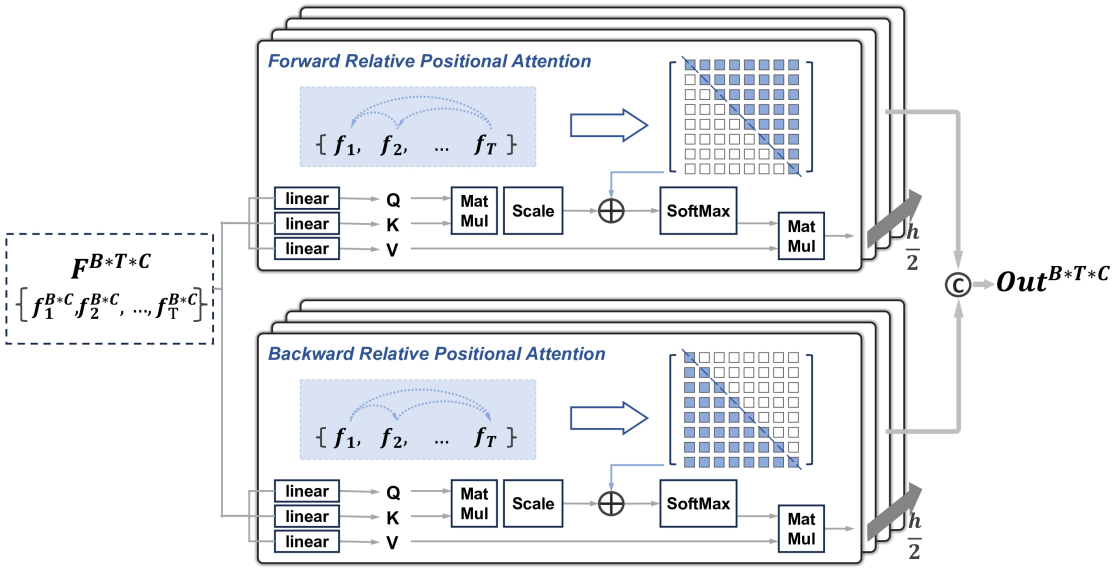


Figure 3: Bidirectional Relative Positional Attention [7]

and a lightweight detection head structure. Ghosh et al. proposed GazeSCRNN, which is a spiking neural network based on a biologically inspired structure, combining convolutional and RNN structures to achieve an efficient temporal modeling, making it particularly suitable for edge scenarios such as AR/VR, where real-time and energy-efficiency requirements are extremely high [16]. In our work, we follow this line of research by proposing a compact framework and conducting related ablation experiments that reduce the inference overhead while maintaining competitive tracking performance.

3 Method

3.1 Overview

Our goal is to design an efficient event-based eye tracking model that achieves competitive prediction accuracy while significantly reducing model complexity and deployment costs.Unlike mainstream architectures (such as CNN + GRU + BRAT) that rely on deep and stacked structures, our approach adopts a simplified design, replacing complex temporal modeling modules with a lightweight Mamba-based sequence model, and employing pruning techniques to reduce the number of parameters in the feature extraction module. The final model has only 265k parameters and achieves a prediction accuracy of 1.65% per error. To highlight the comparison, Figure 4 shows the architectural differences between the top-ranked USTCEventGroup team and our lightweight configuration.

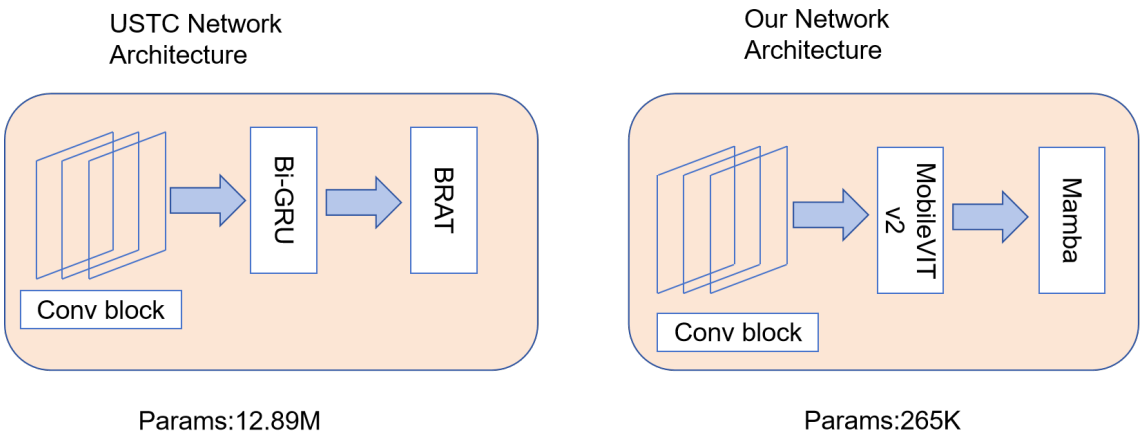


Figure 4: Architecture comparison between USTCEventGroup's model and our proposed lightweight design. On the left is the USTCEventGroup team model architecture, which includes a convolutional feature extraction module, a Bi-GRU time series modeling module, and a BRAT attention head. On the right is our proposed lightweight model architecture, which uses MobileViTv2 to replace the traditional convolutional backbone and uses the Mamba module for time series modeling. This design significantly reduces the number of model parameters while maintaining prediction accuracy.

3.2 Data processing

To convert the original asynchronous event stream into a frame input format which is suitable for feature extraction networks, we designed a preprocessing workflow involving time slicing and weighted accumulation. After this preprocessing, the originally asynchronous event data is divided into a series of more dense frame

sequences based on a fixed event window, which can be directly processed by mainstream convolutional neural networks or temporal models. This method not only preserves the high temporal resolution advantage of event cameras while enhancing the spatial structure advantage of the input data, but also helps downstream models better capture spatio-temporal features.

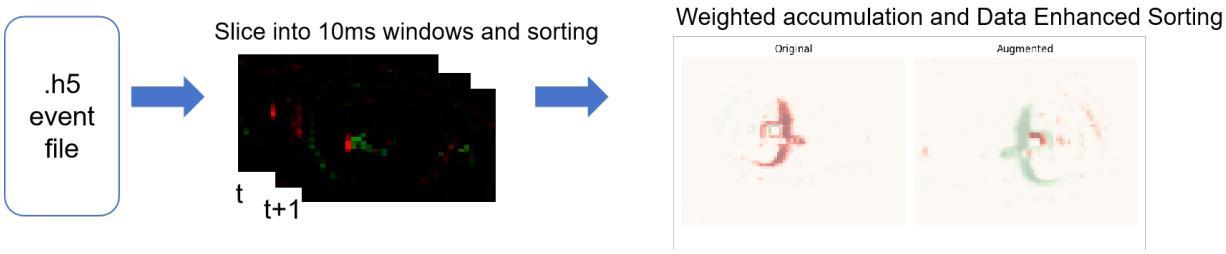


Figure 5: Event Data Preprocessing Flowchart

As shown in the figure 5, this is a framework diagram of a complete data preprocessing workflow. This workflow includes time slicing, polarity map construction, image scaling, and data augmentation through sequence construction.

3.2.1 Time Slicing Accumulation

In order to transformer the event stream into a tensor input with structure, the first step is to temporally order the raw events. We extract all events from the .h5 file, each containing a timestamp (t), spatial coordinates (x, y), and polarity (p, 0 for negative polarity, 1 for positive polarity). Event data is randomly distributed according to the time it is recorded, so it is necessary to sort the data in ascending order by timestamp to ensure that there is no cross-frame interference during slicing. After sorting the data, we slice it into time windows with 10-millisecond increments, dividing the timeline into continuous windows. All events within each time slice are collected and form the base input for that point in time.

3.2.2 Polarity Map and Weighted Accumulation

Each 10ms time window needs to be converted into a tensor frame with the shape $[H, W, 2]$ for subsequent feature extraction. The last dimension contains two channels, which respectively accumulate positive polarity events and negative polarity events. As shown in Figure 6, the green area represents the distribution of positive polarity events, and the red area represents the distribution of negative polarity events.

To construct this tensor frame, all events within the time window are traversed: if an event has positive polarity, its value is accumulated at the corresponding position

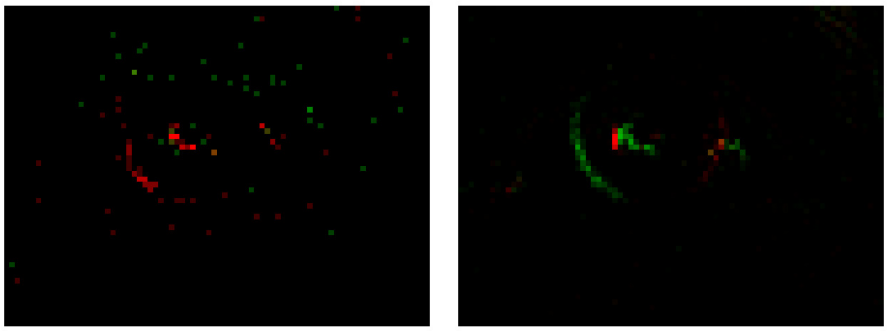


Figure 6: This figure shows a Polarity Map of events within a 10ms time window. The position of each pixel in the image indicates the spatial location of the event triggered in that region, where red represents an ON event (a positive polarity event resulting from an increase in brightness) and green represents an OFF event (a negative polarity event resulting from a decrease in brightness). This approach provides rich spatial contrast information while maintaining temporal polarity.

in the first channel; if negative, it is accumulated in the second channel. To enhance the modeling of short-term dynamics, we introduce a temporal decay weighting strategy. When constructing a multi-frame sequence, multiple adjacent windows are superimposed, and each frame is assigned a weight based on its temporal distance from the current frame—the farther the distance, the smaller the weight—so as to emphasize the event changes occurring at the current moment.

Specifically, let T be the window length, and let Δt be the time difference between a given event and the current frame. The assigned value in the tensor is defined as:

$$I_{x,y,p} = \max\left(I_{x,y,p}, \frac{T - \Delta t}{T}\right) \tag{1}$$

where:

- (x, y) denotes the position of the event on the image.
- $p \in \{0, 1\}$ indicates the event polarity (positive/negative).
- Δt represents the time difference between the event and the current frame.
- A weight value closer to 1 means the event is nearer to the center of the current time window.

This weighting strategy enhances the local structure within the current time slice and improves modeling performance in regions with high motion dynamics.

3.2.3 Caching Strategy

Due to the high resolution of the event camera output (e.g., 640×480), there is a large amount of computational redundancy in direct processing, so we use bilinear interpolation to scale each tensor to 80×60, which significantly reduces the subsequent network load. The scaled tensor still retains the spatial contour features, which can be effectively modeled by the lightweight network.

To avoid repetitive I/O operations and repeated computations, we save each tensor frame in .pt format to disk, and generate corresponding index files and label files (with pupil center locations) according to the dataset division (training/validation/testing), which are convenient for subsequent fast loading and batch training.

3.2.4 Data Augmentation

Considering that the pupil center estimation task is sensitive to spatial distribution and visual microstructure, we adopt several lightweight but effective data enhancement methods:

- Crop: Using the center of the pupil center as a reference, we randomly crop a small area and resize it back to its original size to simulate the viewing area under different focuses.
- Polarity Inversion: Displacing the two polarity channels to improve the model's adaptability to samples with inconsistent polarity.
- Horizontal Flip: mirrors the tensor horizontally with probability 0.5.
- Random Offset (Shift): adds an offset of ±10% pixel range in the X and Y axis directions, respectively.

All enhancement operations are performed with simultaneous label updates to ensure that the viewpoint positions always match the inputs, thus enhancing the generalization ability and robustness of the model.

Figure 7 shows a comparison of the tensor maps before and after enhancement. In the right panel, it can be seen that the offset and flip have clearly occurred, while preserving the key polar features.

3.2.5 Sequence Construction

To implement the timing modeling, we assemble the single-frame tensor into a fixed-length sequence of frames. In this project, the setup is as follows:

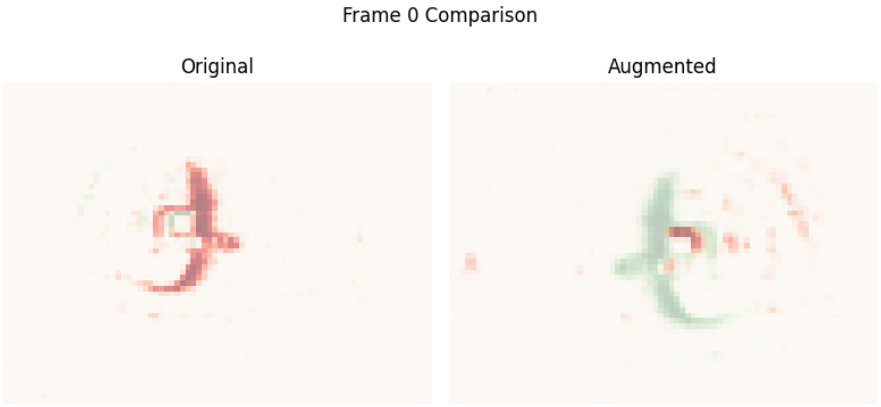


Figure 7: Original polarity map (left) vs. enhanced image (right). Red and green pixels indicate ON and OFF events, respectively. From the figure, it can be observed that the enhanced image undergoes operations such as cropping, polarity flipping, horizontal flipping, and horizontal and vertical displacements, which helps to improve the robustness of the model

- The length of each sequence is 10 frames (T=10).
- The inter-frame interval (stride) is 5 frames.

During construction, the system will backtrack forward with the current frame as the center and collect the history tensor frames to form a four-dimensional tensor of [T, H, W, 2]. If the history is insufficient, the earliest available frame will be copied automatically to make up for it. This structure not only supports the input of temporal models such as GRU/Mamba, but also provides a certain view of history with low memory overhead, which is a proven effective solution in deployment practice. This strategy offers a lightweight yet effective window of temporal context with low memory overhead and has been empirically validated as suitable for real-time deployment scenarios.

3.3 Model Architecture

3.3.1 Overall Design

This study aims to address the task of **eye tracking in event vision scenarios**, i.e., predicting the observer's pupil center coordinates from a dynamic stream of visual events. Due to the asynchronous, sparse, and highly dynamic nature of event data, the model must have both **spatial modeling capabilities** (for capturing the spatial distribution of events) and **temporal modeling capabilities** (for understanding the evolutionary patterns of events over time).

Algorithm 1 Frame Sequence Index Construction

Require: Current frame index i, sequence length T, stride s, file ID list F
Ensure: Frame index sequence S of length T

Initialize $S \leftarrow [\,]$
for $j = T - 1$ to 0 **do**
    $k \leftarrow i - j \cdot s$
    if $k \geq 0$ and $F[k] = F[i]$ **then**
        Append k to S
    else
        Append $S[0]$ to S ▷ Pad with first valid index
    end if
end for
return S

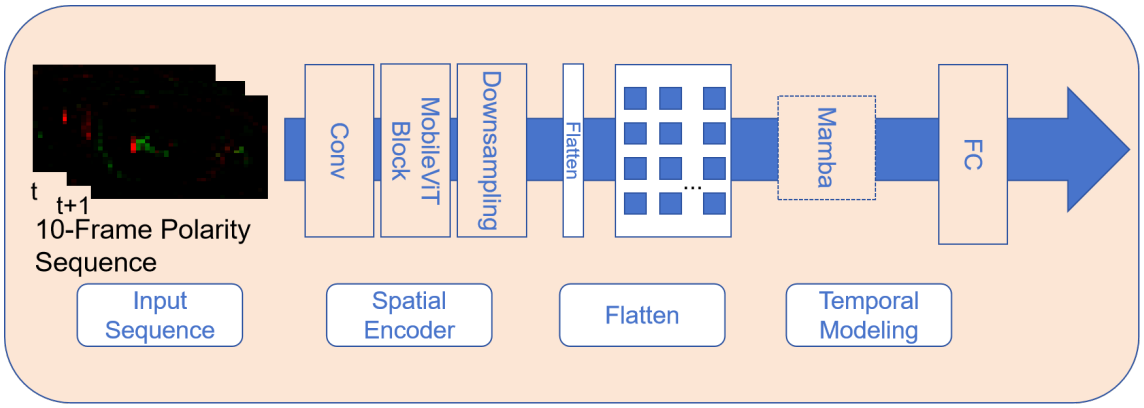


Figure 8: Model Architecture

To this end, we design a lightweight joint spatial-temporal modeling architecture, as shown in Fig. 8. The architecture consists of three main components:

- **Spatial Feature Extractor:** MobileViTv2 is a lightweight visual Transformer architecture designed for mobile devices, serving as an improved version of MobileViT [27]. It achieves significant reductions in model parameters and computational complexity while maintaining high accuracy. When using MobileViTv2 as a spatial encoder, we performed structural pruning to further reduce the number of MobileViT blocks while efficiently extracting features from sparse temporal graphs. This module takes as input a polarity tensor of shape $[H, W, 2]$ for each frame and outputs compressed spatial feature vectors.

- **Temporal Modeling Module:** Mamba is a time series modeling architecture designed to address the high computational complexity and low efficiency of Transformers in long sequence modeling [18]. We chose Mamba as the main time series modeler because it has a state space structure with linear time

complexity and can model long time series dependencies. We also used other time series modeling modules (such as GRU and LAF modules) in our ablation experiments, which are discussed in detail later in this paper.

- **Pupil Center Prediction Layer:** The sequence representation output from the temporal modeling module is passed into a fully connected layer (i.e., a Linear layer), which outputs two normalized floating point numbers representing the x/y coordinates of the predicted pupil center on the image.

The whole process starts from the input event sequence of shape $[T, H, W, 2]$, and then goes through spatial extraction, temporal modeling, and finally coordinate regression to complete the efficient and lightweight pupil center prediction inference process.

3.3.2 Spatial Encoder: Trimmed MobileViTv2

To effectively extract spatial structural information from event sequences, we adopt the MobileViTv2 network as the spatial encoder [32]. MobileViTv2 combines the inductive bias of convolutions with the global modeling capabilities of Transformers, making it highly suitable for processing spatio-temporal visual data. To better accommodate the sparsity and lightweight requirements of event vision tasks, we have customized and simplified the spatial encoder module based on MobileViTv2. This section will first introduce the basic structure of the standard MobileViTv2, followed by a detailed explanation of our proposed pruned version of MobileViTv2 and the modifications and optimizations made to its key modules.

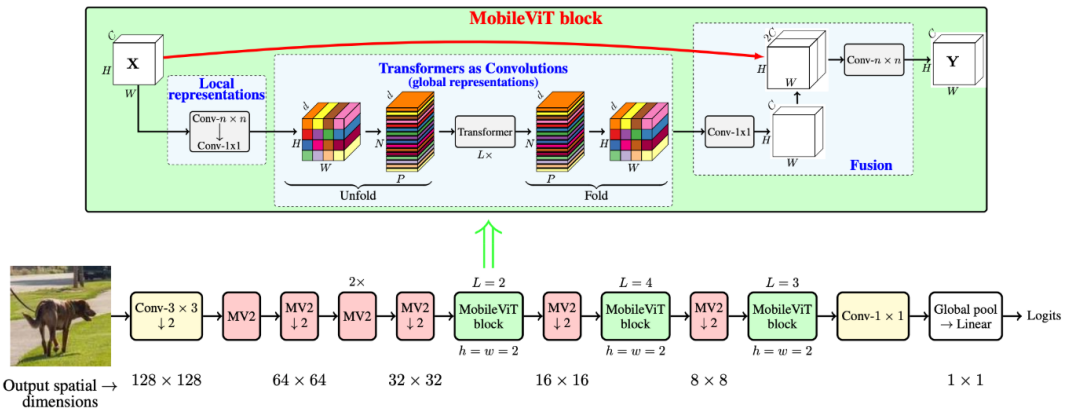


Figure 9: MobileViT structure diagram [27]

standard MobileViTv2 structure Figure 9 shows the overall structure of the MobileViTv2 architecture and the internal components of the MobileViT module, including local convolution layers, patch-based tokenization, transformer modeling, and fusion steps. The main modules of the standard MobileViTv2 include:

- An input adaptation layer: a 3×3 convolution that maps 2-channel polarity input to a 3-channel format compatible with MobileViTv2.
- A Conv2D Stem for basic feature extraction.
- Inverted Residual Block for efficient spatial refinement.
- MobileViT Block for global feature aggregation.

The input event tensor is a polarity map sequence:

$$X \in \mathbb{R}^{T\times 2\times H\times W}$$

After spatial encoding, the output becomes:

$$F \in \mathbb{R}^{T\times C\times h\times w}$$

Where T is the sequence length, C is the output channel, and h, w are the spatial dimensions after downsampling.

(1) Conv2D Stem. Each frame $X_t \in \mathbb{R}^{2\times H\times W}$ is passed through a 3×3 convolution:

$$X_t^{(1)} = \text{Conv2D}(X_t; W_{\text{conv}}, b_{\text{conv}}) \in \mathbb{R}^{C_1\times \frac{H}{2}\times \frac{W}{2}}$$

(2) Inverted Residual Block. The block includes expansion, depthwise convolution, and projection:

$$\begin{aligned} X_{\text{expand}} &= \text{Conv}_{1\times 1}(X_t^{(1)}) \\ X_{\text{dw}} &= \text{DWConv}_{3\times 3}(X_{\text{expand}}) \\ X_{\text{project}} &= \text{Conv}_{1\times 1}(X_{\text{dw}}) \\ X_t^{(2)} &= X_{\text{project}} \in \mathbb{R}^{C_2\times \frac{H}{2}\times \frac{W}{2}} \end{aligned}$$

(3) MobileViT Block. We apply a MobileViT block composed of convolution and Transformer:

$$
\begin{aligned}
X_{\text{local}} &= \text{Conv}_{3\times3}(X_t^{(2)}) \\
\text{Patches} &= \text{Unfold}(X_{\text{local}}, \text{patch\_size} = (p, p)) \\
\text{Tokens} &= \text{Flatten}(\text{Patches}) \in \mathbb{R}^{N_p \times d} \\
\text{Tokens}' &= \text{TransformerLayer}(\text{Tokens}) \\
X_t^{(3)} &= \text{Fold}(\text{Tokens}') \in \mathbb{R}^{C \times \frac{H}{2} \times \frac{W}{2}}
\end{aligned}
$$

This completes the spatial encoding process for a single frame.

Algorithm Explanation. Algorithm 2 outlines the step-by-step spatial encoding process for an event sequence. For each time step t, the corresponding event representation X_t is sequentially passed through the Conv2D stem, the Inverted Residual Block, and the MobileViT block. The resulting spatial feature map $X_t^{(3)}$ is collected and stacked across the temporal dimension to form the final encoded tensor F. This ensures that local patterns and global context are efficiently captured for each frame before temporal modeling.

Algorithm 2 Trimmed MobileViTv2 Spatial Encoder

Require: Event sequence tensor $X \in \mathbb{R}^{T \times 2 \times H \times W}$
Ensure: Spatial feature tensor $F \in \mathbb{R}^{T \times 64 \times \frac{H}{2} \times \frac{W}{2}}$

Initialize Conv Stem, Inverted Residual, and MobileViT Block
for $t = 1$ to T **do**
    $X_t \leftarrow X[t]$ ▷ Shape: $[2, H, W]$
    $X_t^{(1)} \leftarrow \text{Conv2D}(X_t)$
    $X_t^{(2)} \leftarrow \text{InvertedResidual}(X_t^{(1)})$
    $X_t^{(3)} \leftarrow \text{MobileViTBlock}(X_t^{(2)})$
    $F[t] \leftarrow X_t^{(3)}$
end for
$F \leftarrow \text{Stack}(F[1], \ldots, F[T])$
return F ▷ Shape: $[T, 64, H/2, W/2]$

Custom cropping version MobileViTv2 module In order to better adapt to the sparsity and lightweight requirements of event vision tasks, we customized and simplified the spatial encoder module based on MobileViTv2:

- **Lightweight attention mechanism** In MobileViT Block, we use a custom LinearSelfAttention module to replace the standard multi-head self-attention (MHSA). This module generates queries, keys, and values through 1×1 convolutions and uses a non-standard softmax_te activation function for attention normalization, thereby reducing computational complexity and enhancing sparsity and stability. The output is represented as:

$$\text{Output} = \text{ReLU}(V) \cdot \sum(K \cdot \text{softmax\_te}(Q))$$

- **Patch modeling strategy optimization** We adopt an unfold → Transformer → fold processing flow in the MobileViT Block, i.e., first divide the local feature map into patches, then perform global attention modeling, and finally restore the spatial structure. This strategy retains global modeling capabilities while significantly reducing memory and computing requirements.

- **Normalization strategy adjustment** Considering that the batch size of event image sequences is small and unevenly distributed, we uniformly use GroupNorm instead of LayerNorm in the attention module and FFN module to improve training stability.

- **Forward path simplification** Compared with the original MobileViTv2, we only retain the first three layers (Conv Stem + IRB + a set of MobileViT Blocks), with a total output channel count of 64 and a spatial size compressed to half that of the original image, effectively controlling model size and improving inference efficiency.

3.3.3 Temporal Model

Due to the non-uniform, sparse, high-frequency, and asynchronous nature of event data along the temporal axis, effective temporal modeling is a main challenge in event-based vision tasks. Unlike traditional frame-based data, event streams lack fixed frame rates, so it requires models to capture temporal dynamics with low latency and high efficiency. To address this, we designed and compared different temporal modeling structures based on three core modules: Gated Recurrent Unit (GRU), Linear Attention Feedforward (LAF), and the state-space-based Mamba module.

This section provides a detailed description of each temporal module's structure, computation, design motivation, comparison to conventional methods, and its integration in our model architecture.

Gated Recurrent Unit (GRU). GRU [8] is an improved version of recurrent neural networks (RNN), known for its use of gating mechanisms to control information flow and mitigate the vanishing gradient problem. Compared with LSTM, GRU has a simpler structure with fewer parameters and better training efficiency, making it well-suited for low-resource event vision tasks.

The GRU update rules are defined as:

$$
\begin{aligned}
z_t &= \sigma(W_z x_t + U_z h_{t-1}) \\
r_t &= \sigma(W_r x_t + U_r h_{t-1}) \\
\tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1})) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}
$$

where z_t and r_t are the update and reset gates respectively, $\tilde{h}_t$ is the candidate hidden state, and h_t is the final output state. x_t is the input at time t, σ denotes the sigmoid function, and $\odot$ denotes element-wise multiplication.

In our model, features extracted from the MobileViTv2 encoder are globally pooled and flattened into sequences $X \in \mathbb{R}^{B \times T \times D}$, where $D = C \cdot H \cdot W$, which serve as the input to a single-layer GRU. The final hidden state h_T is then used for downstream prediction.

While GRU effectively captures short-term temporal dependencies, it struggles with long-term modeling and lacks explicit mechanisms for selective temporal attention. Therefore, we explore attention-based and state-space alternatives for enhanced modeling capacity.

Linear Attention Feedforward (LAF). To enhance global temporal modeling while avoiding the high computational cost of standard Transformers, we introduce a lightweight attention-based module, LAF [35]. It replaces full softmax attention with a simplified normalization mechanism and combines it with a feedforward network to efficiently model temporal context.

The attention mechanism used in LAF relies on the following activation function:

$$
\text{SoftmaxTE}(q) = \frac{\text{ReLU}(1 + q + 0.5q^2)}{\sum_j \text{ReLU}(1 + q_j + 0.5q_j^2)}
$$

which approximates softmax while ensuring numerical stability and linear complexity. The weighted representations are passed through a two-layer feedforward network with residual connections and normalization.

In our architecture, LAF is stacked after the GRU, enabling the combination of recurrent short-term modeling and attention-based global context encoding. Com-

pared with traditional multi-head Transformer modules, LAF offers:

- Reduced computational and memory overhead.
- Pluggable structure, easily integrated with RNNs or CNNs.
- Stable training due to simplified activation and normalization.

Mamba: State Space Sequence Modeling. Mamba is a recently proposed state space model (SSM) [18] designed to achieve linear-time sequence modeling while maintaining strong global context understanding. Unlike RNNs or Transformers, Mamba models sequence dynamics through discretized continuous-time systems.

The core formulation is:

$$\frac{d}{dt}h(t) = Ah(t) + Bx(t), \quad y(t) = Ch(t) + Dx(t)$$

which can be discretized into a dynamic convolutional operation:

$$y = Ax + B \cdot \text{convolve}(x, K)$$

where K is a learnable kernel generated via interpolation. Mamba supports parallel computation and achieves strong sequence modeling with high GPU efficiency.

Mamba offers:

- Strong long-term dependency modeling;
- Low parameter count and inference latency.
- Efficient GPU execution via parallel computation.
- Superior performance on event-based data.

3.3.4 Output Head and Post-processing

The final prediction head is designed to map the temporal output features into a normalized 2D pupil center coordinate. The output head consists of a single-layer fully connected (FC) projection:

$$\hat{y}_t = \text{FC}(h_T) \in \mathbb{R}^2$$

where h_T is the final hidden state from the temporal module (e.g., GRU, Mamba). The output $\hat{y}_t$ represents the predicted (x, y) pupil center.

Activation Function. In our design, we do not apply a `sigmoid` activation to $\hat{y}_t$. Instead, we allow the network to freely regress over the output range, and apply normalization as part of the post-processing step. This approach preserves gradient dynamics and avoids early saturation. We experimentally verified that adding a sigmoid activation harms convergence and reduces accuracy.

Coordinate Normalization. During training, ground-truth pupil center coordinates are normalized to $[-1, 1]$ range based on the resolution of the original video. The predicted values are then denormalized back to pixel space during evaluation for compatibility with benchmark metrics.

Loss Function and Weighting. We adopt the standard ℓ_2 loss (mean squared error) between predicted and ground truth coordinates. We also experimented with reweighting the loss in high-error samples, but found that uniform weighting yields more stable training. Formally, the loss is:

$$\mathcal{L}_{\text{pupil-center}} = \frac{1}{N} \sum_{i=1}^{N} \|y_i - \hat{y}_i\|_2^2$$

3.4 Training Strategy

Reproducibility and Environment. In order to ensure the stability of the training process and reproducibility of the results, we uniformly set the random seed in each work thread of `NumPy`, `PyTorch`, and the data loader workers. Also, we set `torch.backends.cudnn.deterministic=True` and disable benchmarking to avoid selecting non-deterministic computational kernels. All models were trained on a single NVIDIA GPU using the `PyTorch Lightning` framework, which automatically handles tasks such as device configuration and logging.

Data Loading. We use a preprocessed event dataset to accelerate training, store the processed dataset in `.pt` format, and load it with the customized `H5Dataset` class. The training and validation sets have been partitioned proportionally in advance, and the training is loaded using `DataLoader`. To ensure determinism in the loading process, the training set is set with `shuffle=True` to scramble the data, and all loading threads use `worker_init_fn` based on a fixed random seed. Each sequence sample contains 10 consecutive frames of event images with batch size set to 16.

Learning rate scheduling. According to the learning rate scheduling, we use the current mainstream cosine annealing scheduler (cosine annealing scheduler) [24], which can realize the dynamic adjustment of the learning rate in each training round. The learning rate at the t-th epoch is calculated as:

$$\eta_t = \eta_0 \cdot \frac{1}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right)$$

where $\eta_0 = 1 \times 10^{-3}$ is the initial learning rate and $T = 248$ represents the total number of rounds of training. This scheduling strategy helps the model converge more smoothly and avoids overfitting caused by a fixed learning rate. In our experiment, the cosine annealing scheduling can effectively bring smoother validation loss curves and smaller pixel error fluctuations.

Model saving and logging. We automatically save the models through the `ModelCheckpoint` callback function provided by PyTorch Lightning, which monitors the validation set pixel error (`val_p_err`) and retains the top 30 best-performing models based on the results of the validation set pixel error. The name of the saved model file contains the epoch number and the error metrics, this naming scheme is to facilitate the subsequent evaluation and analysis. The progress and evaluation metrics during the training process are visually recorded and debugged via `TensorBoardLogger`.

4 Experiments

4.1 Experimental Setup

4.1.1 Dataset

This experiment uses the official dataset provided by the 3ET Challenge [7], which contains sequences of incident visual eye-tracking with real pupil center annotations. The dataset is divided into a training set and a validation set. The training set contains 24,788 sequences, validation is performed on a separate subset, and testing is performed on the hidden test set provided by the challenge.

4.1.2 Training Configuration

The optimizer used for model training is Adam, and the initial learning rate is set to 1×10^{-3}, and the cosine annealing strategy is used to gradually reduce the learning rate. The training epoch is set to 248, and the batch size is 16. All experiments

are done on a single NVIDIA RTX 3090 Ti GPU, implemented using the PyTorch Lightning framework, and the training process is monitored by TensorBoard, and the model is saved based on the pixel error of the validation set.

4.2 Evaluation Metric

In order to evaluate the effectiveness of our proposed temporal modeling strategy, we use pixel error as a metric for evaluating the model, with a pixel error of Euclidean distance between the predicted pupil center and the true pupil center locations on an 80×60 downsampled image.

Evaluation Metric. Let $\hat{y}_t \in \mathbb{R}^2$ denote the predicted pupil center at frame t, and $y_t \in \mathbb{R}^2$ be the corresponding ground-truth label. The pixel error is defined as:

$$\text{PixelError} = \frac{1}{N} \sum_{t=1}^{N} \|\hat{y}_t - y_t\|_2 \tag{2}$$

where N is the total number of samples. All coordinates are evaluated in the pixel domain, scaled to the 80×60 spatial resolution used throughout training and testing.

```
       Block Params_K
       conv1    3.23K
       conv2  102.78K
       conv3 1639.94K
          fc   16.58K
         gru 6391.30K
       other    0.00K
transformer 4739.07K

Total Parameters: 12892.90K  (12.89M)
```

Figure 10: Measured Parameter Count of the 3ET Baseline.

4.3 Baseline Reimplementation and Parameter Discrepancy

In previous related work [7], the total number of parameters for this model was reported to be approximately 7 million. However, we couldn't fully reproduce their values because they didn't open-source the implementation code or give us precise ways to count parameters. To verify the actual model size, we reimplemented the model structure based on the description in their paper and automatically counted the parameters for each layer using PyTorch's forward_hook method. As shown in

the figure 10, the final results indicate that the total number of trainable parameters in the model is 12.89 million, significantly higher than the original report. We think that this difference might be because old statistical tools were used or some modules were left out while counting parameters (such GRU or attention structures in Transformers).

4.4 Ablation Study

In order to gain insight into the contribution of the different modules in the proposed model, we performed ablation experiments focusing on two key aspects.

- Comparing different temporal modeling modules (LAF/GRU vs. Mamba).
- Assessing the impact of spatial encoder depth in the MobileViT backbone..

The following Table 1 summarizes their results as measured by pixel error on the test set:

Model	**Temporal Modules**	**#Spatial Layers**	**Params**	**Pixel Error**
0 (3ET Baseline)	GRU + Transformer	3	12.89M	**1.14**
A	GRU + LAF	4	0.84M	1.67
B	GRU + Mamba	4	0.85M	1.63
C	Mamba only	4	0.75M	1.65
D	GRU + LAF	3	0.40M	1.79
E	GRU + Mamba	3	0.41M	1.66
F	Mamba only	3	**0.26M**	1.65

Table 1: Ablation study results for different model variants.

Comparison with Original 3ET Baseline We first compare our models against the official 3ET Baseline (Model 0), which adopts a GRU + Transformer architecture with three spatial encoding layers. Despite its high accuracy (pixel error of 1.14), the large model size poses significant challenges for deployment on resource-constrained platforms.

In contrast, our proposed lightweight variants substantially reduce the parameter count (as low as 0.26M) while maintaining acceptable accuracy. For instance:

- **Model F** (Mamba-only with three spatial layers) achieves a pixel error of 1.65 with only **265K** parameters, reducing model size by approximately **98%** while incurring only 0.51 more error.

- **Model B** (GRU + Mamba with four spatial layers) reaches 1.63 pixel error using just **0.85M** parameters, which is still significantly smaller than the baseline.

These results suggest that our lightweight models are good for situations when resource efficiency is very important.

Temporal Modeling Modules: Comparison between LAF/GRU and Mamba We compare the performance of the traditional GRU + LAF module with the Mamba module under consistent spatial configurations. Results show that replacing LAF + GRU with Mamba **consistently improves model performance**, as summarized below:

- With four spatial layers, **Model B** (GRU + Mamba) outperforms **Model A** (GRU + LAF), reducing pixel error from 1.67 to 1.63.
- Even without GRU, the Mamba-only models (**Model C/F**) maintain a comparable performance to GRU + LAF (pixel error around 1.65).
- Under the lightweight setting (three spatial layers), **Model E** (GRU + Mamba) still outperforms **Model D** (GRU + LAF), demonstrating Mamba's robustness.

These findings confirm that Mamba provides **superior temporal modeling capability and parameter efficiency**, making it an effective alternative to traditional recurrent-attention combinations.

The Role of Spatial Feature Extraction Layer (Layer 4) We also evaluate the impact of the fourth spatial encoding layer (Layer4) in MobileViT. The key observations are:

- Removing Layer4 leads to noticeable performance degradation in both GRU + LAF and GRU + Mamba configurations (Model A $\rightarrow$ D: 1.67 $\rightarrow$ 1.79; Model B $\rightarrow$ E: 1.63 $\rightarrow$ 1.66), highlighting the utility of high-level spatial features.
- However, in the fully lightweight Mamba-only model (**Model F**), removing both GRU and Layer4 still results in a pixel error of **1.65**, which is comparable to the full model (Model A).

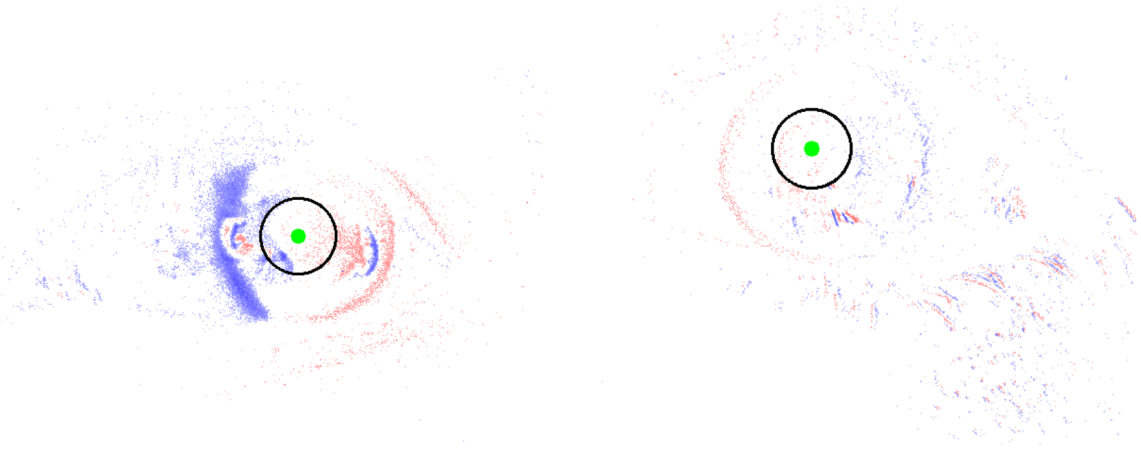
This implies that when Mamba is used, its temporal modeling capacity can **compensate for the absence of deep spatial features** which enable the removal of Layer4 to reduce the parameter count from 841K to 265K with minimal performance loss.

Summary of Findings

Based on the above ablation study, we draw the following key conclusions:

- The Mamba module offers **superior temporal modeling performance** and greater parameter efficiency than the traditional GRU + LAF combination.
- While deeper spatial encoders (e.g., Layer4) contribute positively to prediction accuracy, they can be removed in certain deployment-sensitive conditions.
- Our best model configuration (**three-layer encoder + Mamba-only**) achieves **1.65 pixel error with only 265K parameters**, striking an excellent balance between performance and deployability for edge computing scenarios.

4.5 Visualization



(a) Sample 1 (b) Sample 2

Figure 11: Visualization of predicted pupil center locations (green dots) overlaid on event polarity maps. The black circle indicates the ground truth pupil center area and red and blue represent positive and negative events, respectively.

We present the prediction results for representative samples. The model's predicted pupil center is annotated on the event polarity map which enable us to analyze the model's pupil center prediction behavior more intuitively.The event polarity map is generated by mapping positive-polarity events andnegative-polarity events to red and blue channels, respectively. As shown in Figure 11, the green highlighted areas

indicate the model's expected pupil center positions, i.e., two-dimensional coordinates, while the outer black ring-shaped regions represent the actual pupil center areas.From these two maps, it can be observed that the projected pupil center typically align well with the high-density motion regions around the eyes, indicating that the model can extract important spatio-temporal attributes from sparse event data.

5 Conclusion

In this study, our main goal is to accomplish the task of event-based eye tracking, and to explore whether high prediction accuracy can be achieved without relying on deep and complex structures. Given the increasing demand for real-time and low-power eye tracking in AR/VR applications, we propose a lightweight pupil center prediction model based on a modular spatiotemporal architecture, which effectively combines the MobileViTv2 module for image processing with the Mamba module for temporal modeling. To assess the impact of different architectural designs on performance, we designed a series of different versions model for experimentation, combining different temporal modeling modules (GRU, LAF, and Mamba) as well as different spatial coding depths. Through extensive ablation experiments on the 3ET+ dataset, the results show that the Mamba-based temporal modeling module not only outperforms the traditional GRU + LAF combination in terms of prediction accuracy, but also significantly reduces the number of parameters. Our optimal model achieves a 1.65 pixel error using only 265K parameters, representing over 98% compression in model size compared to the baseline scheme.

The above results validate the feasibility and effectiveness of lightweight structures for eye-tracking tasks in event vision. This result provides a useful reference for subsequent researchers who wish to deploy eye-tracking systems to edge devices.

6 Future Work

Although the approach presented in this paper strikes a good balance between accuracy and efficiency, there are still several promising directions that deserve further exploration:

- **Real-time latency benchmarking on embedded devices.** Deployment on embedded devices. Despite the small parameter size and lightweight structure of our model, it has not yet been deployed and validated on real edge devices (e.g., NVIDIA Jetson, ARM-based SoCs, etc.). The model is still in Float format, and the computational overhead of trying to deploy it on edge devices is a significant issue; future work could try to introduce techniques such as model quantization to further enhance the increased deployability on resource-constrained devices.

- **Online eye tracking with adaptive temporal resolution.** The sliding window of the current model adopts a fixed length, and in the future, we can explore the dynamic window mechanism or online processing framework, so that the system latency can be further reduced and the response speed can be further improved, which is more suitable for streaming input scenarios such as AR/VR.

- **Multi-modal sensor fusion.** Currently we only have a single event data as input, in the future, we can fuse the event data with other modalities (e.g., RGB images, depth sensors, or IMU signals), which is expected to greatly improve the robustness of the model in complex environments (e.g., low light, occlusion, blinking interference, etc.), and realize efficient fusion with lightweight multistream structures.

- **Self-supervised or unsupervised learning.** Current event visual pupil center prediction still relies heavily on manually labeled data. In the future, self-supervised strategies based on motion consistency, contrast learning, etc. can be explored, e.g., we can try to incorporate the optical flow model to reduce the labeling cost and improve the model's generalization ability in unknown environments.

Overall, this study serves as a foundation for efficient and practical event-based pupil center estimation, and we hope to build on this work to develop deployable and adaptive systems for broader real-world applications.

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017.

[2] Anastasios N Angelopoulos, Julien NP Martel, Amit PS Kohli, Jorg Conradt, and Gordon Wetzstein. Event based, near eye gaze tracking beyond 10,000 hz. *arXiv preprint arXiv:2004.03577*, 2020.

[3] Patrick Bardow, Andrew J Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 884–892, 2016.

[4] Rafael Barea, Luciano Boquete, Manuel Mazo, and Elena López. System for assisted mobility using eye movements based on electrooculography. *IEEE transactions on neural systems and rehabilitation engineering*, 10(4):209–218, 2002.

[5] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE transactions on neural networks and learning systems*, 25(2):407–417, 2013.

[6] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[7] Qinyu Chen, Chang Gao, Min Liu, Daniele Perrone, Yan Ru Pei, Zuowen Wang, Zhuo Zou, Shihang Tan, Tao Han, Guorui Lu, et al. Event-based eye tracking. 2025 event-based vision workshop. *arXiv preprint arXiv:2504.18249*, 2025.

[8] Kyunghyun Cho et al. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, 2014.

[9] Tobi Delbruck and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor. *Frontiers in neuroscience*, 7:223, 2013.

[10] Andrew T Duchowski and Andrew T Duchowski. *Eye tracking methodology: Theory and practice*. Springer, 2017.

[11] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[12] Wolfgang Fuhl, Thomas Kübler, Katrin Sippel, Wolfgang Rosenstiel, and Enkelejda Kasneci. Excuse: Robust pupil detection in real-world scenarios. In *Computer Analysis of Images and Patterns: 16th International Conference, CAIP 2015, Valletta, Malta, September 2-4, 2015 Proceedings, Part I 16*, pages 39–51. Springer, 2015.

[13] Wolfgang Fuhl, Thomas Kübler, and Enkelejda Kasneci. Pure: Robust pupil detection for real-world scenarios. In *Proceedings of the 2020 ACM Symposium on Eye Tracking Research and Applications (ETRA)*, pages 1–5. ACM, 2020.

[14] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

[15] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *ICCV*, 2019.

[16] Anirban Ghosh, Wei-Ting Hsiao, Szu-Yu Liu, Pi-Cheng Chen, and Yu-Chiang Frank Cheng. Gazescrnn: Event-based gaze estimation using spiking convolutional recurrent neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. arXiv preprint arXiv:2205.02018.

[17] Arren Glover and Chiara Bartolozzi. Event-driven ball detection and gaze fixation in clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2203–2208. IEEE, 2016.

[18] Albert Gu, Tri Dao, Karan Goel, and Christopher Ré. Mamba: Linear-time sequence modeling with selective state spaces. In *International Conference on Learning Representations (ICLR)*, 2024.

[19] Khadija Iddrisu, Waseem Shariff, and Suzanne Little. A framework for pupil tracking with event cameras. In *26th Irish Machine Vision and Image Processing Conference (IMVIP)*. IET, 2024.

[20] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European conference on computer vision*, pages 349–364. Springer, 2016.

[21] Jongmin Lee, Youngwook Kim, Dongyoon Kim, and Seungryong Lee. Deep learning-based pupil center detection for fast and accurate eye tracking system. *Electronics*, 10(13):1532, 2021.

[22] Jun Haeng Lee, Tobi Delbruck, Michael Pfeiffer, Paul KJ Park, Chang-Woo Shin, Hyunsurk Ryu, and Byung Chang Kang. Real-time gesture interface based on event-driven processing from stereo silicon retinas. *IEEE transactions on neural networks and learning systems*, 25(12):2250–2263, 2014.

[23] Yulan Liang, Michelle L Reyes, and John D Lee. Real-time detection of driver cognitive distraction using support vector machines. *IEEE transactions on intelligent transportation systems*, 8(2):340–350, 2007.

[24] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[25] Päivi Majaranta and Kari-Jouko Räihä. Text entry by gaze: Utilizing eye-tracking. *Text entry systems: Mobility, accessibility, universality*, (2007):175–187, 2007.

[26] Mehrube Mehrubeoglu, Linh Manh Pham, Hung Thieu Le, Ramchander Muddu, and Dongseok Ryu. Real-time eye tracking using a smart camera. In *2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–7. IEEE, 2011.

[27] Sachin Mehta. Mobilevitv2: Improved mobile-friendly vision transformer with simple and effective fusion, 2022.

[28] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. Hfirst: A temporal approach to object recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):2028–2040, 2015.

[29] Wei-Liang Ou, Tzu-Ling Kuo, Chin-Chieh Chang, and Chih-Peng Fan. Deep-learning-based pupil center detection and tracking technology for visible-light wearable gaze tracking devices. *Applied Sciences*, 11(2):851, 2021.

[30] Bharath Ramesh, Shihao Zhang, Zhi Wei Lee, Zhi Gao, Garrick Orchard, and Cheng Xiang. Long-term object tracking with a moving event camera. In *Bmvc*, page 241, 2018.

[31] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2016.

[32] Mohammad Rastegari Sachin Mehta. Mobilevitv2: Light-weight and general-purpose vision transformers. *arXiv preprint arXiv:2206.02680*, 2023.

[33] Amos Sironi, Marco Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *CVPR*, 2018.

[34] Lech Świrski, Andreas Bulling, and Neil Dodgson. Robust real-time pupil tracking in highly off-axis images. In *Proceedings of the symposium on eye tracking research and applications*, pages 173–176, 2012.

[35] Ashish Vaswani et al. Attention is all you need. In *NeurIPS*, 2017.

[36] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 2018.

[37] Zhong Wang, Zengyu Wan, Han Han, Bohao Liao, Yuliang Wu, Wei Zhai, Yang Cao, and Zheng-jun Zha. Mambapupil: Bidirectional selective recurrent model for event-based eye tracking. *arXiv preprint arXiv:2404.12083*, 2024.

[38] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. It's written all over your face: Full-face appearance-based gaze estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2291–2300, 2017.

[39] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *arXiv e-prints*, pages arXiv–1802, 2018.