Automated Segmentation and Classification of Eye Movements in a Depth-Varying Robotic Workspace

by

Reshma Seby John

(Under the Direction of Deborah Barany and Suchendra M. Bhandarkar)

Abstract

Accurate detection of eye movements is critical for analyzing visuomotor behavior. However, automated classification methods have not been thoroughly explored for data collected using depth-varying stimuli within a shared hand-eye workspace. Using data obtained from an augmented-reality robotic setup with integrated monocular remote eye tracking, we present a two-stage deep learning pipeline for automated eye movement classification. The pipeline uses Improved Naive Segmented Linear Regression (INSLR) for signal denoising and segmentation, followed by a bidirectional long short-term memory (BiLSTM) model that classifies segments into fixations, saccades, and smooth pursuits. This method demonstrates a sample-level accuracy of 88.0%, outperforming other modern eye movement detectors. The approach provides a reliable framework for automated gaze detection, and addresses challenges that existing methods struggle with in depth-varying environments.

Index words: Eye Movement Classification, Bidirectional LSTM, Piecewise Linear Segmentation, Gaze Event Detection, Time Series Segmentation

Automated Segmentation and Classification of Eye Movements in a Depth-Varying Robotic Workspace

by

Reshma Seby John

B.E., Birla Institute of Technology and Science Pilani, 2021

A Thesis Submitted to the Graduate Faculty of the

University of Georgia in Partial Fulfillment of the Requirements for the Degree

Master of Science

Athens, Georgia

2025

Automated Segmentation and Classification of Eye Movements in a Depth-Varying Robotic Workspace

by

Reshma Seby John

Major Professors: Deborah Barany

Suchendra M. Bhandarkar

Committee: Fei Dou

Electronic Version Approved:

Ron Walcott

Dean of the Graduate School

The University of Georgia

December 2025

Acknowledgments

I am deeply grateful to my major professors, Dr. Deborah Barany and Dr. Suchendra M. Bhandarkar, for their guidance and expertise throughout this work. They taught me a lot about turning technical ideas into careful research, and I'm grateful for the space they gave me to explore new ideas. Their feedback has helped me grow as a researcher. I am also thankful to my committee member, Dr. Fei Dou. Learning the fundamentals of data mining from her made a big difference for me, and her feedback has strengthened this thesis. I would like to thank my friends for always showing up and checking in. Their support kept me grounded and moving forward. Finally, my deepest gratitude to my parents and my sister for their love, motivation, and for always making time to listen. Their faith in me has been my constant foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Eye tracking has become a fundamental tool in various fields, including neuroscience, cognitive psychology, human-computer interaction, and clinical research (Klaib et al., 2021). It allows researchers to study visuomotor coordination by assessing visual attention during hand-eye coordinated tasks, and can reveal impaired visual processing through abnormal eye movement patterns (Cheng et al., 2023). Eye tracking can also help detect early signs of neurological problems, such as Alzheimer's and Parkinson's disease (Tokushige et al., 2023, Pavisic et al., 2017, Culicetto et al., 2025). Modern eye-tracking systems record gaze data with high temporal and spatial resolution. They capture continuous eye movements as participants follow visual stimuli. To extract meaningful information and make valid deductions from these recordings, raw gaze signals must be classified into distinct eye movement events. The three major eye movements are fixations, saccades, and smooth pursuit. Each of these reflects different aspects of oculomotor control and visual processing.

Manually labeling eye movements is time-consuming and subject to variability between annotators, especially for ambiguous events such as post-saccadic oscillations and fixation-pursuit transitions. Human labelers may find smooth pursuit events difficult to label without additional target information. Ambiguous transitions between certain eye movements, such as fixations and slow pursuits, remain challenging even among expert raters (Agtzidis et al., 2016). The labeling process becomes even more difficult when the gaze data is noisy. Hence, automated classification of these events is essential for reducing the labor and subjectivity associated with manually evaluating and classifying the various eye movements in the data.

Over the years, researchers have developed various algorithms for classifying eye movements, ranging from manual feature engineering to more recent machine learning and deep learning approaches (Andersson et al., 2017, Birawo and Kasprowski, 2022). The recent deep learning approaches, such as a one-dimensional Convolutional Neural Network combined with a Bidirectional Long Short-Term Memory (1D CNN-BLSTM) model (Startsev et al., 2019) and Temporal Convolutional Networks (TCN) (Elmadjian et al., 2021, Elmadjian et al., 2023), have demonstrated strong performance on benchmark datasets, such as GazeCom (Dorr et al., 2010). These models generally outperform earlier feature-based methods that rely on manually tuned thresholds. Deep learning models use end-to-end learning to classify eye movements directly from raw or minimally preprocessed gaze coordinates. They achieve high sample-level and event-level accuracy on data recorded in controlled laboratory environments. However, most of these models were developed and evaluated on fixed-depth, screen-based recordings. They struggle when applied to depth-varying data from systems such as the KINARM (KINARM, Kingston, Ontario, Canada). The KINARM system integrates a monocular remote eye tracker with a robotic arm workspace and is widely used

in sensorimotor neuroscience to study naturalistic eye-hand coordination. In augmented-reality and robotic systems such as the KINARM, visual targets appear at varying depths within a shared hand-eye workspace in the transverse plane (Singh et al., 2016). The Naive Segmented Linear Regression and Hidden Markov Model (NSLR-HMM) framework proposed by Schrayer, 2022 was specifically designed for KINARM data, but its classification accuracy leaves room for improvement in practical research applications.

1.2 Motivation

The limitations of existing classification methods on KINARM data directly motivated this work. Although existing deep learning approaches have achieved strong results on benchmark datasets, these models were developed for two-dimensional recordings where visual stimuli remain at a fixed depth in the frontal plane. However, the KINARM uses 2D gaze tracking in the transverse plane within peripersonal space, where targets appear at varying depths. The integrated EyeLink 1000 (SR Research Ltd., Ottawa, Ontario, Canada) eye tracker records at 500 Hz. This improves temporal resolution but also makes the raw gaze traces noisier than those from lower-frequency systems. Because visual targets appear at varying depths, the KINARM recordings contain an additional vergence component. Vergence is the inward and outward rotation of the eyes to maintain focus at different depths or distances (Mahanama et al., 2022, Purves et al., 2001). Due to this, the KINARM data differs fundamentally from data captured on a fixed 2D screen (Singh et al., 2016). Unlike most of the benchmark datasets, the KINARM data is pursuit-rich, meaning it contains a higher proportion of smooth pursuits. The model must learn to find fixations and saccades within

these pursuit-heavy sequences. Also, the recordings contain movements that are difficult to classify, such as microsaccades within fixations, catch-up saccades during smooth pursuit, post-saccadic oscillations following saccades, and ambiguous boundaries between fixations and slow pursuits. Noise from individual differences in oculomotor behavior also makes classification harder, because the models must learn to adapt to these variations. Standard time-series filtering techniques, such as low-pass Butterworth and Savitzky-Golay filtering, do not fully resolve these issues. Over-smoothing removes genuine high-velocity components such as saccades, while under-smoothing leaves noise that degrades classifier performance. Due to all of these factors, models trained on fixed-depth datasets do not generalize well to KINARM data. The unique characteristics of the gaze behavior require alternative classification strategies.

Automated eye movement classification requires dividing the continuous gaze data into temporal segments and assigning each segment to a class based on features such as angular velocity and displacement. This can be framed as a time-series segmentation problem, where accurate segmentation is important because downstream research depends on proper boundary placement and effective noise handling. The need for a method that can segment and classify eye movements automatically from KINARM gaze data influenced the direction of this thesis. Motivated by these challenges, a segmentation-first approach combined with deep learning classification was investigated as a way to handle the pursuit-rich, noisy recordings while adapting to subject-specific differences. This work also explored an adapted online approach as an alternative model for improving real-time classification performance on KINARM recordings.

1.3 Contributions

The primary aim was to build an automated pipeline for classifying fixations, smooth pursuits, and saccades for depth-varying, pursuit-rich KINARM recordings. For this, a preprocessing workflow tailored to the KINARM data was developed. This includes a two-pass blink and missing-data detector that finds both the central missing-data area and the noisy shoulders around each blink. It then passes these ranges to the later preprocessing code for removal. Together with minimal filtering and angular velocity calculation, these steps produce preprocessed gaze sequences that can be used directly for segmentation, model training, and evaluation.

The main contribution of this thesis is the INSLR-BiLSTM model pipeline, which combines Improved Naive Segmented Linear Regression (INSLR) (adapted from Johari et al., 2024) with a Bidirectional Long Short-Term Memory (BiLSTM) deep learning classifier. INSLR is used to denoise and segment the gaze signal into piecewise-linear segments. For each segment, a set of kinematic and spatial features is computed. Then, a BiLSTM uses a short context window over neighboring segments to assign each segment to one of the three classes. This decoupled, segmentation-first design is intended to handle the pursuit-rich, noisy KINARM recordings more effectively than fully sample-based models, and to provide interpretable segment-level features that can be reviewed if needed.

As a second contribution, an adapted online TCN model was developed for real-time use. Unlike the original online TCN by Elmadjian et al., 2023, which relies on multi-scale features computed from causal context windows, the adapted TCN works directly with per-sample features and uses causal dilated convolutions with a receptive field of about one second of past context. A

minimum fixation-duration constraint is built into the training to reduce label interleaving during noisy fixations. This model is intended for per-sample, online classification of KINARM data when real-time detection is needed.

Both models were evaluated on the same KINARM dataset and compared against established baselines. The INSLR-BiLSTM and adapted TCN models were tested using sample-level and event-level metrics. The INSLR-BiLSTM had 88.0% overall sample-level accuracy and performed the best out of all the tested models. The adapted TCN reached 81.1% overall sample-level accuracy, performing better overall than the online TCN model by Elmadjian et al., 2023. As an additional check, the INSLR-BiLSTM model was also tested on the GazeCom dataset (Dorr et al., 2010) to assess transfer to fixed-depth, screen-based recordings, where it achieved performance comparable to recent eye movement detection research.

1.4 Outline of Thesis

The thesis is organized as follows:

Chapter 2 reviews background on eye movements, eye tracking technology, and algorithms for signal denoising, segmentation, and classification, including deep learning methods like BiLSTM and TCN.

Chapter 3 describes the KINARM dataset, preprocessing steps, the proposed INSLR-BiLSTM model combining segmentation with BiLSTM classification, an adapted online TCN model, baseline methods, and evaluation protocols using sample-level and event-level metrics.

Chapter 4 presents results on the KINARM dataset, with detailed performance analysis. The results of the exploration on the GazeCom dataset are also presented.

Chapter 5 concludes the work by discussing contributions, limitations, and future directions that can be explored to improve upon this work.

Chapter 2

Literature Review

2.1 Eye Movements

2.1.1 Fixations

Fixations occur when the gaze fixates on a specific location, allowing visual processing. The optimal minimum duration threshold for fixations has been identified as 60 ms. This is particularly true in tasks involving reading and visual search where fixations are usually shorter than during scene perception tasks (Trabulsi et al., 2021, Rogers et al., 2018). Angular velocities during fixation typically stay below 30°/s (Trabulsi et al., 2021, Rogers et al., 2018).

Even during fixations, the eyes are not perfectly stationary. They exhibit small, involuntary movements, such as microsaccades, ocular drift, and tremors (Ko et al., 2016, Mahanama et al., 2022). These fixational eye movements introduce noise in the gaze data. But they serve a critical function by preventing visual fading that occurs when the same image remains on the retina for too long (Engbert, 2006). Microsaccades have been shown to counteract perceptual fading by refreshing the retinal image when objects begin to fade from view (Costela et al., 2013, Krauzlis et al., 2017,

Martinez-Conde et al., 2006). The fovea is the small central area of the retina responsible for sharp, detailed vision. As long as drift and other movements do not carry the gaze substantially outside the foveal region, it is still considered to be part of a single continuous fixation. Accurate detection of fixations is essential for understanding visual attention and information processing during tasks such as reading and visual search.

2.1.2 Smooth Pursuit

Smooth pursuits enable the eyes to follow moving targets while maintaining the target's image on the central visual field (Purves et al., 2001). The minimum duration threshold for smooth pursuit is 40 ms (Singh et al., 2016). Smooth pursuit movements typically have angular velocities ranging from 20-40°/s, but can go higher depending on the task (Mahanama et al., 2022).

The ability to smoothly follow a moving target deteriorates when target velocities exceed 30°/s, so catch-up saccades are often needed to maintain tracking (Nachmani et al., 2020, Heinen et al., 2016). This occurs because, during smooth pursuit, the eye may lag behind the target due to delays in visual processing and sensorimotor control. When this positional lag becomes significant, catch-up saccades are triggered to bring the target back onto the fovea. Catch-up saccades can happen when pursuit gain is low or due to unexpected changes in the velocity or direction of the stimuli (de Brouwer et al., 2002). In this work, we classify catch-up saccades as saccades rather than as part of the smooth pursuit movement.

2.1.3 Saccades

Saccades are rapid eye movements that shift gaze between locations (Birawo and Kasprowski, 2022, C.-M. Wang and Hsu, 2024). The minimum duration threshold for saccades is 5 ms (Singh et al., 2016). Depending on the method and task, reported minimum saccade angular velocities are usually between 30°/s and 100°/s (Birawo and Kasprowski, 2022, Guadron et al., 2024, Glaser et al., 2020, Schütz et al., 2014).

Post-saccadic oscillations (PSOs) are corrective, damped oscillatory movements that can occur at the end of saccades (Del Punta et al., 2019). PSOs are caused by the overshoot of the eyeball beyond its intended target position and represent the mechanical settling of the eye following the rapid movement. Some eye movement classification algorithms treat PSOs as separate events distinct from saccades. However, in this work, we consider PSO as part of the saccadic movement, since it does not consistently occur after all saccades (Del Punta et al., 2019).

2.1.4 Blinks

Blinks occur naturally during regular eye movements and are the closing and opening of eyelids (Mahanama et al., 2022). A blink consists of three distinct phases: the closing phase when the eyelids shut, the closed phase when the eyelids are fully closed, and the opening phase when the eyelids reopen (Nyström et al., 2024). This means that a blink is not just the period of missing data during full eyelid closure, but also the entire eye movement process from closing to opening, which typically lasts between 150 and 400 ms (Nyström et al., 2024).

Some eye movement classification algorithms treat blinks as separate events and classify them as a distinct eye movement category (Elmadjian et al., 2023). However, in this work, blinks are detected and removed from the data, instead of being classified as a separate eye movement type. This approach allows the focus to remain on analyzing the actual eye movements of interest, specifically fixations, saccades, and smooth pursuits.

2.1.5 Eye Tracking Technology

Eye tracking systems record gaze position and eye movements using different measurement approaches. The three main categories are video-based optical tracking, electrooculography (EOG), and eye-attached tracking with scleral coils (Klaib et al., 2021). Video-based systems are the most widely used, and they rely on infrared illumination and cameras to track the pupil center and corneal reflection. These systems enable non-invasive recording at high temporal and spatial resolution. The video-based systems can be set up as screen-based setups, where the camera is positioned at a distance from the participant.

Benchmark datasets, such as GazeCom (Dorr et al., 2010), were recorded using remote eye trackers at a sampling rate of 250 Hz, with participants viewing video stimuli at fixed distances on a frontal display. In contrast, the data collected from the KINARM (KINARM, Kingston, Ontario, Canada) have fundamentally different characteristics. The KINARM is a remote monocular system integrated with the EyeLink 1000 (SR Research Ltd., Ottawa, Ontario, Canada), recording at 500 Hz, with stimuli presented on a horizontal plane in the transverse (peripersonal) workspace. The system outputs two-dimensional gaze point-of-regard (POR) coordinates and three-dimensional gaze vectors (X, Y, Z) for the eye. The gaze POR represents the point on the display surface

where the participant is looking, calculated by tracking the position of the pupil and a reflection from the cornea using infrared light (Mestre et al., 2018, Guestrin and Eizenman, 2006). The three-dimensional gaze vectors are components of a unit vector that represents the direction of the line of sight from each eye in a camera-centered coordinate system, which can be used to estimate depth and vergence information. This configuration introduces depth variation as visual stimuli move closer to or farther from the participant, requiring careful consideration of vergence and pursuit dynamics during classification.

2.2 Algorithms

2.2.1 Signal Denoising

Signal denoising, or filtering, reduces unwanted variability in time series data while preserving relevant features. For eye movement data, denoising is necessary because raw gaze position traces contain noise from eye movements and measurement imprecision from pupil tracking errors. Without denoising, velocity calculations become unreliable and classification algorithms struggle to distinguish event boundaries from high-frequency noise.

Common filtering methods for gaze data include several approaches (Raju et al., 2021, Warman et al., 2017, Valliappan et al., 2020). Savitzky-Golay filtering fits a low-order polynomial to a sliding window to smooth the signal (Savitzky and Golay, 1964). Butterworth low-pass filters attenuate high-frequency components beyond a chosen cutoff frequency (Butterworth et al., 1930). Median filters replace each point with the median of its neighbors (Harezlak and Kasprowski, 2019). Moving average filters average neighboring samples within a window (Chauhan et al., 2018). Gaussian

filters convolve the signal with a Gaussian kernel to suppress noise (Špakov, 2012). Other methods, such as Kalman filtering (Kalman, 1960) and Wiener filtering (Mejia-Romero et al., 2021), have also been applied to gaze signals.

A critical challenge in gaze data filtering is balancing noise reduction against the preservation of legitimate high-velocity components. The goal is to attenuate noise such as microsaccades and tremor within fixations, while leaving saccades intact. Over-smoothing can cause saccadic signals to leak into adjacent fixations or smooth pursuits. This leakage can lead to curved transitions where boundaries should be sharp. This makes event boundaries harder to detect accurately and can lead to misclassification of saccade onsets and offsets. On the other hand, under-smoothing may fail to remove the noise that the filter is intended to suppress. Selecting appropriate filter parameters is particularly difficult for KINARM data, where pursuit-rich gaze patterns and catch-up saccades within smooth pursuits introduce additional complexity. The edge-preserving method, anisotropic diffusion, which is used in image analysis and adapted to 1D time series data, is discussed in the following section. It offers a potential solution by selectively suppressing noise while maintaining sharp edges at saccade boundaries.

2.2.2 Anisotropic Diffusion and Adaptation to 1D Time Series

Anisotropic diffusion is an edge-preserving smoothing technique that selectively reduces noise while keeping sharp boundaries (Weickert, 1998). This technique is commonly used in image processing to smooth noise while preserving object edges. It uses a diffusion coefficient that varies depending on the local gradient magnitude. Diffusion is suppressed at saccade edges and is stronger in relatively homogeneous regions. For 1D gaze time series data, the method is adapted by

computing two finite-difference gradients at each time point i (to the left and right neighbors) as $\Delta_N = I[i-1] - I[i]$ and $\Delta_S = I[i+1] - I[i]$. Each gradient is then mapped through a decreasing function of its magnitude:

$$c(\Delta) = \exp\left(-\frac{\Delta^2}{\kappa^2}\right)$$

where κ is a threshold parameter that controls edge sensitivity. The signal is updated using a weighted sum of the diffusion terms, and this update is repeated for a small number of iterations with a stable step size γ. In this way, discontinuities such as saccade onsets are preserved. It tends to reduce high-frequency noise within fixations and smooth pursuits while preserving large discontinuities at saccade boundaries, provided that saccade-related gradients are substantially larger than those associated with noise.

This approach is used in this work because it smooths noise in fixations and smooth pursuits while preserving the sharp boundaries of saccades. By choosing a value for κ that corresponds to typical saccade gradients, the diffusion coefficient becomes small at saccade boundaries. This helps prevent the smoothing from spreading across event transitions. The implementation in this work follows these steps with tunable parameters for the iteration count, conduction coefficient κ, and step size γ.

2.2.3 Piecewise Linear Segmentation

Time series segmentation divides a continuous signal into discrete temporal intervals. For eye movement data, piecewise linear approximation divides the gaze signal into straight-line segments, with each segment representing a part of the gaze trajectory.

Naive Segmented Linear Regression (NSLR), introduced by Pekkanen and Lappi, 2017, combines signal denoising with segmentation by fitting a piecewise linear function to the gaze coordinates. The algorithm produces segments of varying length that together form a piecewise linear approximation of the gaze signal. Both x and y gaze positions are segmented at the same time points, so the resulting segments describe the gaze trajectories. The piecewise linear fit essentially produces a denoised version of the gaze signal. The algorithm uses dynamic programming to find an efficient segmentation, with parameters that control the number of segments created and the expected level of noise in the data.

Improved Naive Segmented Linear Regression (INSLR), by Johari et al., 2024, extends NSLR by modifying how segmentation candidates are selected. During segmentation, the algorithm progresses through the gaze signal and generates multiple competing hypotheses for where each segment should end. In the original NSLR, each candidate boundary is evaluated individually, and the highest-scoring option is selected at each step. This can be problematic in noisy regions where many candidate boundaries appear within a short time window. The noise-driven candidates can break a single eye movement into several short segments.

INSLR addresses this by grouping hypotheses with similar endpoints into clusters and then evaluating these clusters together, rather than comparing each hypothesis individually. The clustering

is performed on boundary times using a fast one-dimensional k-means procedure. In practice, candidate boundaries that occur close together in time are clustered, and isolated boundaries that fall far from the main cluster or form only small groups are penalized. This explicit penalty reduces the influence of isolated candidates on the final segmentation. This step ensures that only boundary splits that are well supported are considered, which helps prevent over-fragmentation in regions where the signal is noisy. The result is a more robust segmentation that performs better on data recorded in real-world settings. Like NSLR, INSLR produces a sequence of linear segments rather than individual data points, allowing a classifier to operate on segments instead of processing each recorded sample separately. Both NSLR and INSLR were paired with Hidden Markov Models (HMM) by Pekkanen and Lappi, 2017, Schrayer, 2022, and Johari et al., 2024. An adapted version of INSLR is used in this work as part of the segmentation module, which is described in Chapter 3.

2.2.4 Bidirectional Long Short-Term Memory (BiLSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network designed to process sequential data (Hochreiter and Schmidhuber, 1997). It keeps an internal memory state that carries information across time steps. LSTMs have been applied to a variety of eye movement analysis tasks. These models leverage the ability of LSTMs to capture temporal dependencies in sequential gaze data.

Bidirectional Long Short-Term Memory (BiLSTM) extends the standard LSTM by processing sequences in both the forward and backward directions simultaneously (Graves and Schmidhuber, 2005). This bidirectional structure consists of two LSTM layers, each reading the sequence from opposite directions. Their outputs are combined at each time step. This allows the network to use

both past and future context when making predictions, which improves classification accuracy for tasks where the whole sequence provides important information. BiLSTM layers have been used in the 1D CNN-BLSTM model for eye movement classification (Startsev et al., 2019).

LSTM and BiLSTM architectures have been widely adopted for classification tasks that operate on short windows of time-series data or on fixed-length feature vectors. In human activity recognition, CNN-LSTM networks have been used to classify sliding windows of accelerometer and gyroscope data into discrete activity categories (Khatun et al., 2022). BiLSTM models have similarly been applied to human activity recognition tasks. For example, a Deep SE-BiLSTM (Jameer and Syed, 2023) was used for classifying short temporal windows of wearable sensor data into activity classes. Also, CNN-LSTM architectures have shown high accuracy on eye-tracking data for autism spectrum disorder diagnosis, using convolutional layers followed by an LSTM layer to classify fixed-length gaze feature vectors (Al-Adhaileh et al., 2025). These applications show that LSTM and BiLSTM-based architectures can learn from compact temporal inputs, such as short sensor windows or fixed-length gaze feature vectors, and can output class predictions directly without requiring additional post-processing layers.

2.2.5 Temporal Convolutional Network (TCN)

Temporal Convolutional Networks (TCNs) are convolutional neural networks designed to process sequential time-series data using dilated causal convolutions. TCNs have been used for eye movement event classification on long temporal sequences, where a non-causal TCN model classifies fixations, saccades, and smooth pursuits from gaze data (Elmadjian et al., 2021). More recently, causal TCN architectures have been adapted for real-time ternary eye-movement classification in an online

setting (Elmadjian et al., 2023). This lets the model look back over a larger time window in the gaze signal without adding much computational overhead. Overall, TCNs are well suited to eye-movement detection because they can capture long-range temporal patterns in gaze data while remaining efficient to train and run.

2.3 Models for Eye Movement Detection

2.3.1 Existing Models

Eye movement classification methods have evolved from simple threshold-based algorithms to sophisticated deep learning architectures. These approaches can be grouped into four main categories, each addressing limitations of earlier methods while introducing new challenges.

Velocity and Dispersion Threshold-Based Approaches

Velocity and dispersion threshold-based methods represent one of the earliest classes of automated eye movement detection algorithms (Salvucci and Goldberg, 2000). The I-VT (Velocity Threshold Identification) algorithm classifies samples as fixations or saccades based on point-to-point velocity, with fixations identified when velocity falls below a threshold (typically 30-100°/s). The I-DT (Dispersion Threshold Identification) algorithm examines the dispersion of gaze points within a sliding temporal window. Windows whose dispersion stays below a chosen threshold are grouped and labeled as fixations. Hybrid algorithms, such as I-VVT (Velocity-Velocity Threshold) and I-VDT (Velocity-Dispersion Threshold) (Komogortsev and Karpov, 2013), combine velocity and dispersion criteria to classify fixations, saccades, and smooth pursuits. REMoDNaV (Robust Eye

Movement Detection for Natural Viewing) (Dar et al., 2021) builds on adaptive velocity-based approaches by using robust statistics and by splitting long recordings into shorter segments with more uniform noise levels. Even though these methods are computationally efficient and more interpretable, they depend on manual threshold tuning that does not generalize across recording conditions. As a result, slow pursuits are often misclassified as fixations.

Probabilistic Approaches

Probabilistic approaches use statistical models to describe eye movements and the uncertainty in their labels over time. I-BDT (Bayesian Decision Theory Identification) (Santini et al., 2016) applies Bayesian decision theory to eye movement classification by using simple gaze features to distinguish between eye movements. The GMM-HMM (Gaussian Mixture Model-Hidden Markov Model) (Xie et al., 2025) combines Gaussian mixture models with a Hidden Markov Model to classify eye movements in an unsupervised and low-latency way. The NSLR-HMM approach (Schrayer, 2022, Pekkanen and Lappi, 2017) uses Naive Segmented Linear Regression to first segment the gaze signal. Then it applies a Hidden Markov Model to classify the resulting linear segments. INSLR-HMM (Johari et al., 2024) follows the same idea but replaces NSLR with Improved NSLR, so that hypothesis clustering during segmentation can improve reliability on noisy, real-world data. These probabilistic methods can be more robust to noise and can encode likely transitions between events, but HMM-based models depend strongly on the chosen state structure and feature design, which can limit how well they transfer across different datasets and recording conditions.

Machine Learning Approaches

Recent machine learning approaches for eye movement classification use ensemble and tree-based classifiers trained on hand-crafted feature representations. Zemblys et al., 2018 introduced the Identification by Random Forest (IRF) method. This uses a random forest classifier trained on features such as velocity and dispersion to classify raw gaze samples into fixations, saccades, and post-saccadic oscillations. The Cascade Forest (EMCCF) (C. Wang et al., 2024) method uses layered ensembles of tree-based classifiers, with each layer refining predictions from previous layers. These ensemble models can perform well even when training data are limited and are relatively easy to interpret, but they still rely on hand-crafted feature sets and do not learn temporal dependencies in the same way as recurrent networks.

Deep Learning Approaches

Recent work in eye movement classification uses deep learning models that learn features directly from gaze data. The 1D Convolutional Neural Network and Bidirectional Long Short-Term Memory (1D CNN-BLSTM) architecture (Startsev et al., 2019) combines convolutional layers for local feature extraction with bidirectional LSTM layers for temporal modeling of short gaze sequences. Temporal Convolutional Network (TCN) models (Elmadjian et al., 2021, 2023) use stacks of dilated convolutions to process sequential gaze data with a large receptive field. The offline TCN (Elmadjian et al., 2021) applies non-causal convolutions to fixed one-second windows of precomputed speed and direction features. The online TCN model (Elmadjian et al., 2023) uses a causal architecture with four temporal blocks so that predictions at time t depend only on samples up to t. The CNN-Transformer model (Suhari et al., 2023) combines a convolutional front-end with Transformer-based

attention over the feature sequence. EmMixformer (Qin et al., 2024) integrates adaptive filtering with a Transformer backbone and mixed attention mechanisms. These deep learning approaches learn useful representations directly from the gaze signal, rather than relying on hand-tuned thresholds and rules. They have been reported to perform well on datasets with noisy signals and mixed event types.

2.3.2 Performance Evaluation and Datasets

Most existing methods focus on three main eye movement classes: fixations, saccades, and smooth pursuits. Some also detect and classify post-saccadic oscillations, blinks, and undefined samples. Performance is usually assessed using sample-level and event-level precision, recall, and F1 scores.

Most benchmark datasets used in modern eye movement classification research consist of recordings made in fixed-depth setups, with stimuli presented on a frontal plane parallel to the participant's viewing direction. Recent models have been evaluated primarily on the GazeCom dataset (Dorr et al., 2010) (250 Hz, 18 video clips, 47 observers per clip). This dataset contains monocular recordings of participants viewing dynamic videos on a vertical screen positioned 45 cm in front of them.

2.3.3 Method-Based Gaps

Fixations and slow smooth pursuits can have similar velocity and dispersion values, especially in pursuit-rich datasets. This makes them hard to separate reliably. Also, small gaze displacements that introduce noise within fixations (microsaccades, drift, and tremor) can be mistaken for smooth pursuit or saccade movements. These issues highlight a key weakness of simple velocity and

dispersion threshold methods, and point to the need for more flexible models that can use additional features and context.

Standard Hidden Markov Model (HMM)-based methods represent eye movements as transitions between a few latent states with Gaussian emission distributions over segment-level features. When fixation and pursuit segments occupy overlapping regions in this feature space, the HMM may confuse these two classes, even though saccades remain easier to detect.

Chapter 3

Methodology

3.1 Hardware and Software Details

The majority of preprocessing, training, and evaluation were performed on an Ubuntu 22.04.4 LTS workstation with an Intel Core i9-10980XE (18 cores/36 threads) CPU and an NVIDIA RTX A5000 GPU. Experiments used Python 3.12.11 with PyTorch 2.8.0+cu126 (CUDA 12.6 runtime).

3.2 Dataset

3.2.1 Dataset Details

The dataset used in this work was collected by Schrayer, 2022 and consists of eye-tracking data recorded during tasks in the KINARM End-Point Lab (KINARM, Kingston, Ontario, Canada). The KINARM system, shown in Figure 3.1, integrates a remote monocular EyeLink 1000 eye tracker (SR Research Ltd., Ottawa, Ontario, Canada) positioned behind the robot, recording gaze data at 500 Hz. Visual stimuli were presented on a horizontal display in the transverse plane within the

participant's peripersonal workspace, with a locking chair used to stabilize the participant's head position during recording.

Figure 3.1: A research participant seated at the KINARM End-Point Lab with a monocular EyeLink 1000 eye tracker. Reprinted from Barany et al., 2020.

Participants tracked white circular targets (1 cm diameter) that moved along five trajectory types, as shown in Figure 3.2: horizontal, vertical, diagonal, sinusoidal, and pseudorandom paths. The target behavior was varied during each trial. This included static events where the target remained stationary to elicit fixations, continuous events where the target moved smoothly along the trajectory to elicit smooth pursuits, and disjoint events where the target jumped ahead to elicit saccades. Trajectories of all types were included in each block of 25 trials presented in random order. Trials ranged from approximately 5 to 30 seconds in length.

Data from four right-handed participants (female) with no history of visual, neurological, or skeletomuscular problems was used for this work.

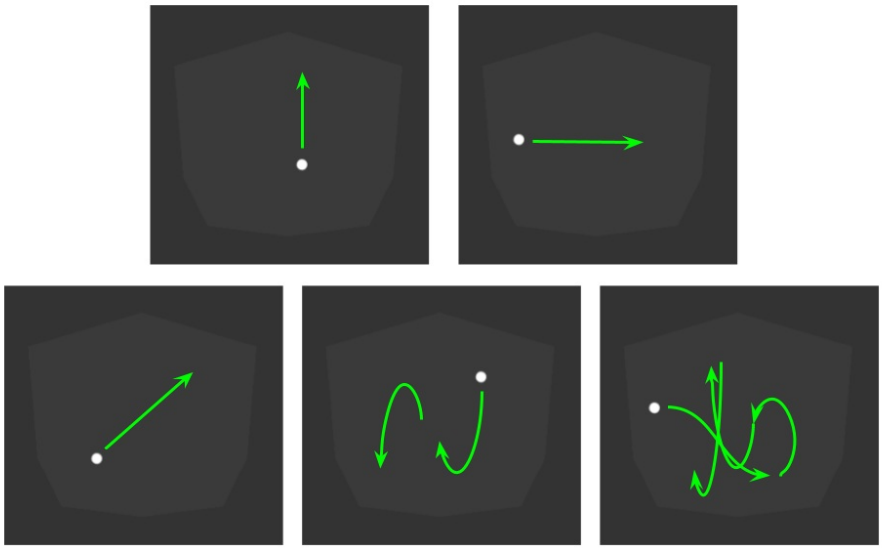


Figure 3.2: Sample trajectories of each type - vertical, horizontal, diagonal, sinusoidal, and pseudorandom - used to elicit fixations, smooth pursuits, and saccades. Reprinted from Schrayer, 2022.

3.2.2 Dataset Features

Gaze data are stored in C3D (Coordinate 3D) files, which is a binary format used for motion capture data. While the KINARM system records multiple gaze features, this work uses the fields listed in Table 3.1. The C3D files were visualized using Dexterit-E Explorer, the KINARM data analysis software. An example of the raw data visualized in Dexterit-E Explorer with Gaze_X and Gaze_Y values plotted is shown in Figure 3.3.

Table 3.1: Gaze data features used from KINARM recordings

Feature	Units	Description
Gaze_X, Gaze_Y	meters	Two-dimensional gaze position on the display
Gaze_Vector_X, Gaze_Vector_Y, Gaze_Vector_Z	unitless	Components of the three-dimensional unit vector for gaze direction
Gaze_TimeStamp	seconds	Timestamp of each recorded gaze sample

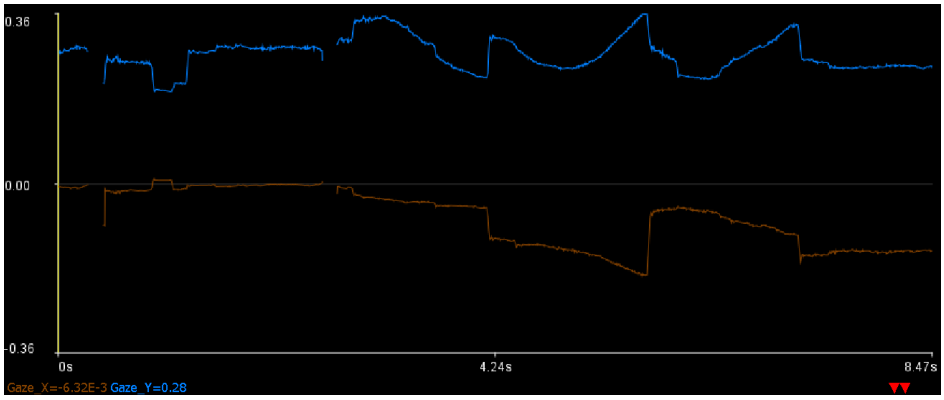


Figure 3.3: Raw gaze data from a trial displayed in Dexterit-E Explorer. The visualization shows unfiltered gaze position (x, y) over time.

The dataset is constituted by 22.81% fixations, 35.46% saccades, and 41.73% smooth pursuits. The average durations were 549.9 ms for fixations, 589.1 ms for smooth pursuits, and 30.8 ms for saccades.

3.2.3 Manual Labeling of Data

The data was manually labeled by Schrayer, 2022 using five categories: Fixations (F), Saccades (S), Smooth Pursuits (P), Blinks (B), and Missing Data (M). The labeling was done by recording the start time in milliseconds of each eye movement within each trial. In this setup, each eye movement is

assumed to end when the next one begins. Before analysis, the labeled data were carefully reviewed to double-check and ensure the highest possible accuracy and consistency with the eye movement definitions discussed in the literature review. This was especially important for confirming that microsaccades, drifts, and tremors remained part of fixations, that catch-up saccades were labeled as saccades rather than smooth pursuits, and that post-saccadic oscillations were always included within the saccadic event.

One of the primary challenges when checking the manual labels was placing precise boundaries between fixations and slow smooth pursuits because determining the exact time point at which one event ended and the next began was often ambiguous. A second difficulty was handling noisy fixations and deciding whether slow drift had moved the gaze sufficiently far outside the foveal region that it should no longer be treated as part of the same continuous fixation. These difficulties are not specific to human annotators, as automated detection methods also struggle with them. However, the detection methods can at least apply the same underlying decision criteria consistently across the dataset.

3.2.4 Handling Blinks and Missing Data

Missing data occurs randomly in the dataset, either when the eye tracker occasionally fails to track the participant's eye or when the participant keeps their eye closed during a trial. Periods of noisy data, where the gaze position is unstable and does not match any recognizable eye movement pattern, are also grouped into this missing data category. Blinks form a separate label and present a particular challenge because they are noisy, ambiguous events that are accompanied by missing data. The inherent ambiguity of these two non-movement classes made it difficult to obtain clean examples

for the models to learn from. Consequently, blinks and missing data were labeled and subsequently removed from the dataset, as the focus of this work is on classifying the eye movements themselves.

To support the overall goal of automated eye movement detection, a preliminary module for blink and missing data detection was developed. Since the model only classifies fixations, smooth pursuits, and saccades, this module identifies the time ranges containing blinks and missing data and passes them to the preprocessing code so that those samples can be removed from the dataset. The module proved especially helpful for blink detection, as it captures not only the periods of missing data during full eye closure, but also the noisy regions before and after the blink (see Figure 3.4), which are critical for proper removal.

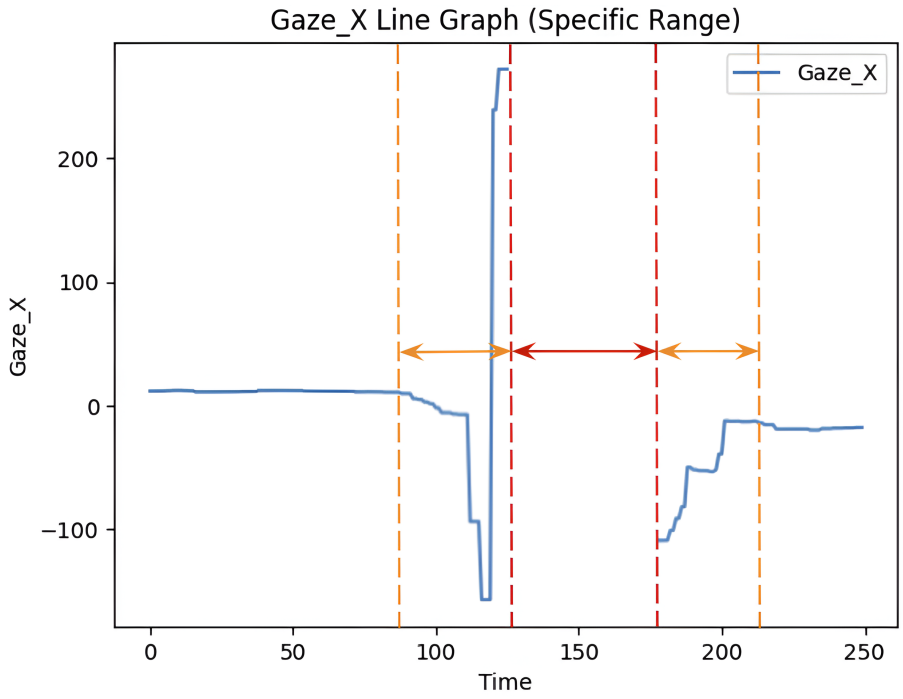


Figure 3.4: Example of blink in gaze X position. The middle segment shows the area of missing data and the adjacent noisy shoulders precede/follow the blink (all three are detected by the two-pass detector).

Detection in the preliminary blink and missing data module was carried out in two passes. In the first pass, the module finds regions of missing data, including the missing samples that occur during blinks. The second pass is then used to identify the noisy shoulders of the blinks.

- Step 1 (anchor the core): Locate periods where the eye was not tracked. In the EyeLink output, these are marked by extreme placeholder values (Gaze_X = Gaze_Y = -100m and Gaze_Vector_X/Y/Z = +100). These blocks form the core of each missing data and blink. This is enough for missing data, but for blinks, it is also necessary to find and include the noisy shoulders around the core. For that, a second step is performed.

- Step 2 (grow the edges with a Gaze-Change Index, GCI): A simple gaze-change index is calculated as a rolling average of point-to-point changes in Gaze_X and Gaze_Y (window = 5). This highlights the brief dip/rebound and jitter that typically occur just before and after a blink. The calm/active threshold is set automatically for each trial (median + (0.5 $\times$ MAD)) so that the method adapts to different noise levels without manual tuning. From each core edge, the process expands outward until encountering a short burst of activity (at least 2 samples above the threshold) followed by a sustained return to calm (at least 6 samples below it). The final mask, therefore, covers the missing-data plateau and its surrounding noise. These mask indices are stored and exported so that they can be used to mark the blinks/missing data for removal during data preprocessing.

3.3 Preprocessing

3.3.1 Data Preparation

Raw KINARM exam (.zip) files containing C3D data were loaded using a Python-MATLAB bridge interface. The MATLAB analysis scripts provided by KINARM were called from Python to extract the exam data into a temporary MAT-file. This was then loaded into Python using `scipy.io.loadmat`. The data was parsed into a `KinarmExam` object that structures the trial data, metadata, and gaze calibration information.

For each trial, the raw gaze position data (Gaze_X and Gaze_Y) and gaze vector components (Gaze_Vector_X, Gaze_Vector_Y, Gaze_Vector_Z) were extracted. Invalid gaze values, marked by extreme placeholder values (Gaze_X = Gaze_Y = -100m and Gaze_Vector_X/Y/Z = +100), were replaced with NaN because they correspond to missing data samples that are removed in later preprocessing steps. Valid gaze positions were scaled from meters to millimeters by multiplying by 1000.

3.3.2 Signal Denoising and Filtering

To denoise the gaze data, a two-stage filtering approach was applied. First, an adapted 1D anisotropic diffusion filter (explained in Section 2.2.2) was applied to the two gaze position components and the three gaze vector components. The anisotropic diffusion filter was configured to run for 10 iterations, a conduction coefficient (kappa) of 50, and a gamma (integration constant) of 0.2.

Following anisotropic diffusion, a Savitzky-Golay filter was applied to all gaze signals with a window length of 10 samples and a polynomial order of 2. This smoothing step reduced residual noise after the initial adaptive filtering.

3.3.3 Feature Calculation

Angular velocity represents the rate of change in gaze direction over time. It is critical for distinguishing between eye movement types because fixations, smooth pursuits, and saccades have characteristic velocity ranges. Following the approach of Schrayer, 2022, angular velocity was calculated from consecutive unit gaze vectors using the arccosine of their dot product:

$$\omega_i = \frac{\arccos\left(\mathbf{G}_i \cdot \mathbf{G}_{i-1}\right)}{\Delta t}$$

where $\mathbf{G}_i = (G_{x,i}, G_{y,i}, G_{z,i})$ is the unit gaze vector at time point i, and $\Delta t = 0.002$ seconds is the sampling interval (since the sampling rate is 500 Hz). The angle returned by the arccosine is first obtained in radians and then converted to degrees, so that ω_i is expressed in degrees per second. Angular velocity was computed only for samples where both $\mathbf{G}_i$ and $\mathbf{G}_{i-1}$ contained valid (non-NaN) values. For samples where the preceding gaze vector value was invalid or missing (including the first valid sample in each trial and any sample immediately following a run of missing data), the angular velocity cannot be computed directly, so its value was set equal to that of the following sample to maintain data length. This feature is a primary cue for both segmentation and classification, since it directly reflects the kinematic differences between eye movement types.

3.3.4 Further Data Preparation

After manual labeling, only the labeled trials were used for further processing and supervised learning. Within each labeled trial, the data was split at the points marked as blinks or missing data, resulting in sequences that contain one or more contiguous eye movement events and exclude any non-movement intervals. Each sequence retained its associated event labels, and sequences with fewer than three samples were discarded to ensure meaningful segments for analysis. This approach ensures that the model is trained only on continuous periods of valid eye movement data.

Additionally, to align the temporal resolution of the dataset, the gaze data were downsampled by a factor of two. This step was necessary because the EyeLink 1000 eye tracker samples at 500 Hz, but the KINARM records at 1000 Hz. This resulted in each gaze entry being duplicated in the dataset, and downsampling eliminated these duplicates. Each downsampled data point was assigned a label corresponding to the eye movement event at that timestamp. The final dataset consists of labeled sequences with three classes: Fixation (F), Smooth Pursuit (P), and Saccade (S). The labeled sequences represent blocks of eye movement data, each containing one or more events. They are then used for subsequent model training and evaluation.

3.4 Proposed Model - INSLR-BiLSTM

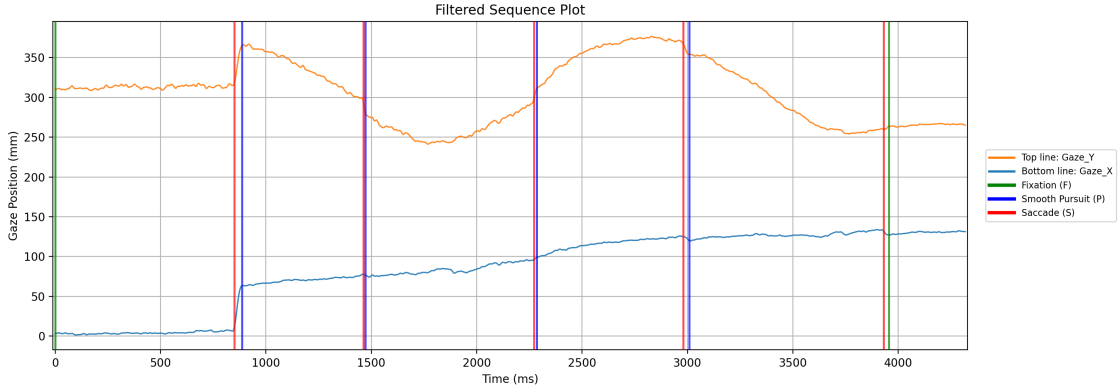
3.4.1 Improved Naive Segmented Linear Regression (INSLR)

The Improved Naive Segmented Linear Regression (INSLR) implementation used in this work is adapted from the NSLR segmentation pipeline described by Schrayer, 2022. Schrayer modified Pekkanen and Lappi's original NSLR code (Pekkanen and Lappi, 2017) to process KINARM gaze data, making domain-specific adjustments for noise and sequence boundaries. Building on her foundation, further changes were introduced for this study to incorporate the INSLR approach by Johari et al., 2024, which improves robustness in the segmentation of real-world eye movement data. The conceptual details of NSLR and INSLR, including hypothesis generation and clustering, have already been described in Section 2.2.3. Here, the focus is on how this implementation was adapted to the KINARM dataset.

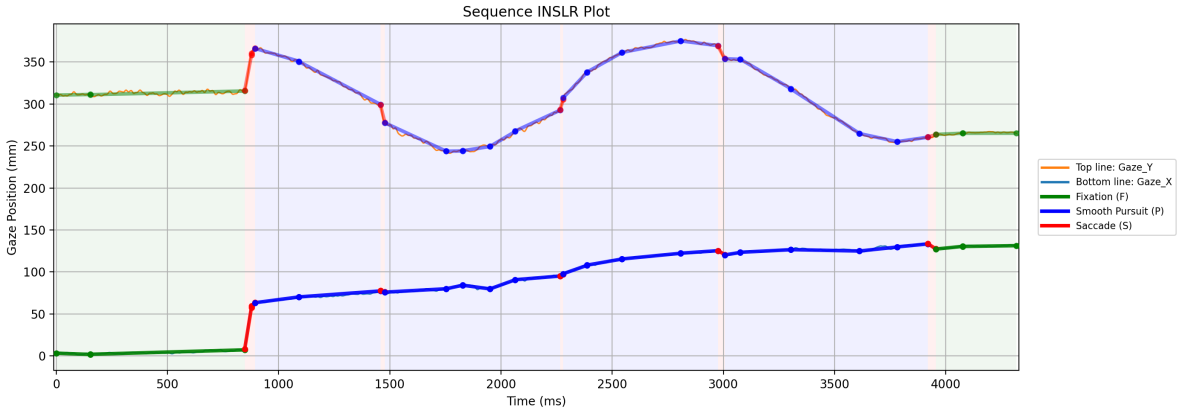
Table 3.2: INSLR parameter values used for the KINARM dataset

Parameter	Value	Description
Clustering Frequency	Every 20 samples	Hypothesis clustering interval (approximately every 40 ms at 500 Hz)
Maximum Clusters	8	Upper limit on clusters per hypothesis set
Cluster Count Heuristic	Number of candidate boundaries ÷ 4	Dynamic cluster count adjustment, reduced when there are fewer candidate boundaries (bounded 1-8)
Outlier Penalty Scale	0.5 × split-likelihood magnitude	Penalty strength for boundaries outside main clusters, scaled relative to split likelihood magnitude
Structural Error	3	Standard-deviation multiplier for noise estimation in INSLR iterations

Several parameters were adapted for the characteristics of the KINARM dataset, including how frequently clustering is run, limits on the number of clusters allowed per hypothesis set, and the size of the penalty used to discourage outlier boundaries. These have been detailed in Table 3.2. Additional code updates ensure that segments have reasonable minimum and maximum lengths, and improve error logging to flag unstable regions in the data. Throughout, the INSLR code continues to segment both gaze dimensions (x and y) at shared boundary points. This is consistent with prior NSLR applications and the needs of downstream sequence modeling. Figure 3.5 shows an example of INSLR applied to a gaze sequence with all three eye movement classes.



(a)



(b)

Figure 3.5: (a) Filtered input sequence. (b) INSLR piecewise-linear reconstruction with shared X/Y boundaries.

Overall, the adapted INSLR implementation provides a stable piecewise-linear representation of the KINARM gaze data, even in regions with non-uniform noise. This segmentation forms the foundation for the subsequent feature calculations and event labeling steps described in the next sections.

Feature Calculations for INSLR Segments

After INSLR segmentation produces piecewise linear approximations of the gaze signal, a set of features is calculated for each segment to characterize its kinematic and spatial properties. We use speed and direction related features because modern algorithms suggest that these features are the most useful for classification. We do not use acceleration features since these have shown less clear benefit in this context, especially for fixation and smooth pursuit detection. These features are extracted from both the segmented linear fits and the underlying raw gaze data, and serve as input to the BiLSTM classifier. They are organized into several categories based on their type and purpose.

Spatial Features - describe the location and extent of each segment:

- Segment endpoints (horizontal) (mm): The x coordinates at the beginning and end of each segment.
- Segment endpoints (vertical) (mm): The y coordinates at the beginning and end of each segment.
- Horizontal and vertical displacement (mm): The absolute change in gaze position along each dimension, computed from segment endpoints.

- Euclidean displacement (mm): The straight-line distance between the segment's start and end points, showing how far the gaze moved during that segment.
- Displacement angle (deg): The direction of motion from segment start to end, measured in degrees.

Temporal and Kinematic Features - capture motion characteristics:

- Cartesian speed (mm/s): The Euclidean displacement divided by segment duration, expressed in millimeters per second, representing the average linear speed of the gaze over the segment.
- 75th percentile angular velocity (deg/s): The 75th percentile of angular velocity values within the segment. Using the 75th percentile rather than the mean provides a more robust measure that avoids the influence of maximum angular velocity values that may be outliers. This feature is useful for distinguishing between eye movement types, as average values alone can be similar across fixations and slow smooth pursuits or between smooth pursuits and saccades.
- Combined speed metric: A feature derived by combining the 75th percentile angular velocity with the Cartesian speed. This composite measure helps the model better differentiate between the three eye movement classes.

Contextual Features - encode sequence information:

- First/last segment indicator: A binary flag marking whether the segment is the first or last in a sequence, signaling that no prior or future temporal context is available.

Each of these features is scaled using standardization (zero mean and unit variance) based on statistics computed from the training set to ensure stable model training.

Labeling INSLR Segments

After INSLR produces segments, each segment must be assigned a class label corresponding to one of the three eye movement types (fixation, smooth pursuit, or saccade). This is done using a labeling scheme adapted from Schrayer, 2022, which applies a plurality voting approach to the underlying sample-level labels that fall within each segment. The dominant class within the segment is determined using a conditional rule based on a saccade share threshold.

Saccade classification: If the proportion of saccade samples within the segment exceeds a threshold, the entire segment is labeled as a saccade. This is important for transitional portions within longer segments that contain a mixture of saccade and other eye movements. In natural eye movement data, saccades often contain transitional segments at their boundaries, where other eye movement types are present, such as between a saccade and a following smooth pursuit. This should ideally be split into separate segments by INSLR, but may occasionally remain as a single segment. By using the threshold, segments that tend to have higher angular velocities and are part of the saccadic event are correctly classified with the saccade rather than being misclassified with adjacent eye movements like smooth pursuits or fixations.

Fixation versus Smooth Pursuit classification: For segments that do not meet the saccade threshold, the dominant class between Fixation and Smooth Pursuit is determined by plurality voting. The class with more samples is assigned to the segment. If the two classes have equal representation, the first occurrence in the segment determines the label.

The primary difference from Schrayer's approach is the saccade threshold value. This work uses a threshold of 15%, compared to the 10% threshold used in prior work. The increased threshold provides slightly higher robustness against misclassification and can be adjusted depending on the characteristics of the dataset.

3.4.2 Bidirectional Long Short-Term Memory (BiLSTM)

After INSLR segmentation and feature extraction, a Bidirectional Long Short-Term Memory (BiLSTM) network is used as a segment-level local-context classifier to classify each segment into one of three eye movement classes: fixation, smooth pursuit, or saccade. Unlike traditional LSTM-based classification approaches that operate on raw time-series windows, the BiLSTM architecture used in this work processes segment-level feature representations within a short window of neighboring segments and predicts the class of the middle segment. As described in the previous section, each segment is characterized by a compact feature vector derived from the piecewise linear fit and underlying gaze signal properties. The BiLSTM does not operate on the individual samples within each segment. Instead, all information is provided through these segment-level feature vectors and their local context.

Classifying at the segment level rather than at the individual-sample level offers several advantages. Sample-by-sample classification in noisy data often produces interleaving of labels and over-segmentation, where single continuous eye movements are incorrectly fragmented into multiple short segments. This requires a post-smoothing method, which may not work efficiently with the fragmentation. In contrast, the combination of piecewise linear segmentation, segment-level features, and local-context BiLSTM classification operates at a higher structural level, using neighboring

segments to help label each segment. This approach better captures the underlying organization of eye movement events while keeping the segment labels stable and interpretable.

Implementation

The BiLSTM model is implemented using PyTorch and is trained using the Adam optimizer with cross-entropy loss. The full pipeline is shown in Figure 3.6, from segmentation and windowing, to the two-layer BiLSTM, and the time-distributed fully connected head.

Input Representation

Each gaze sequence produces a variable-length series of segments that are fed to the BiLSTM. Before being input to the network, segment features undergo preprocessing as follows:

- First, logarithmic scaling is applied to duration, Cartesian speed, and 75th percentile angular velocity to compress their dynamic range.
- Second, a centered rolling window of width 3 is applied, concatenating features from the previous segment, current segment, and next segment into a single feature vector. For boundary segments, zero-padding is used when adjacent segments are unavailable.
- After windowing, each segment is represented by a 36-dimensional feature vector (12 base features $\times$ 3 temporal positions). This corresponds to the boxes before the BiLSTM in Figure 3.6. Note the $K = 3$ context and zero-padding at sequence boundaries.

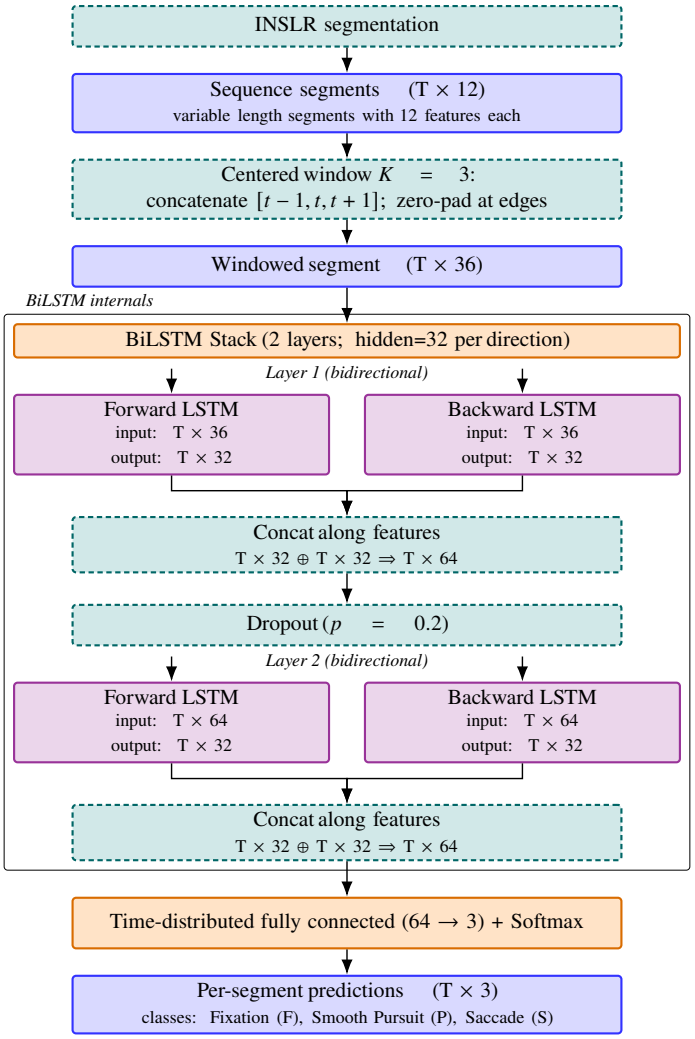


Figure 3.6: BiLSTM diagram: BiLSTM for segmented input: 36-dimensional segment features from a centered K=3 window produce per-segment probabilities for fixation, smooth pursuit, and saccade.

A context window of $K = 3$ was selected after trying a few alternatives. With $K = 1$ (no context), performance on boundary segments dropped, while $K = 5$ tended to over-smooth the labels and blur short saccades. Using three segments gave enough local context without washing out brief events.

Network Architecture

The BiLSTM model consists of a compact two-layer architecture:

- BiLSTM layer: A bidirectional LSTM with 32 hidden units per direction (64 total) and 2 stacked layers. Dropout regularization of 0.2 is applied between layers.
- Projection layer: A fully connected linear layer maps the 64-dimensional BiLSTM output to 3 class logits for each segment.
- Output: Softmax activation produces class probabilities for each segment.

Within each batch, sequences are padded to a common length, and a binary mask is used to ignore padded positions during loss computation and evaluation. The model predicts labels for all segments in a sequence.

Training Configuration

The model is trained using the Adam optimizer with a learning rate of 0.001 and default PyTorch parameters. Cross-entropy loss is computed across all non-padded segment positions, with the ignore index set to -1 for masked positions. Training proceeds for 35 epochs with a batch size of 16 sequences. Data are split at the sequence level (80/20 split) to ensure that segments from the same sequence do not appear in both the training and test sets. Feature standardization is applied using z-score normalization based on statistics computed from the training set only. No class weighting was applied during training, allowing the model to reflect the natural distribution of eye movement classes present in the data.

The bidirectional architecture allows the network to incorporate context from both preceding and following segments when classifying each segment. This is valuable for resolving ambiguous transitional segments at boundaries between different eye movement types.

3.5 Online Model - Adapted Online Temporal Convolutional Network (TCN)

An adapted Temporal Convolutional Network (TCN) was also developed as a possible online alternative to the TCN model described by Elmadjian et al., 2023. One of the main differences is that the model by Elmadjian et al., 2023 uses windows to process the data, while the model here does not use any windowing. In this model, causal dilated convolutions produce a label at each time step based only on past and current data, maintaining an online capability without explicit windowing. In addition, the model is made fixation-aware by using a duration constraint that discourages fixations that are shorter than the minimum fixation duration of 60 ms (30 timesteps at 500 Hz). This was added because the TCN had a tendency to cause interleaving labels, especially during noisier fixations, creating short false saccade or pursuit blips inside. Enforcing a minimum fixation length reduces these and makes the labels more stable and realistic.

The motivation for developing this adapted TCN was to create a model that can be used as an online model that could give quick predictions at every time point. Initial experiments suggest that it offers some improvement overall, especially in saccade detection.

Feature Calculation

Prior to TCN classification, additional feature extraction was performed to augment the raw x and y gaze coordinates and capture further movement-relevant features:

- Deviation from initial position: For each sequence, gaze coordinates are converted into movements relative to the starting point. This reduces location bias, so the model focuses on movement patterns instead of fixed screen positions.
- Cartesian speed features: Cartesian velocities along the x-axis and the y-axis are computed for each sample, providing information about the speed and direction of eye movements.
- Angular velocity: The per-sample gaze angular velocity calculated during preprocessing is kept to strengthen motion cues alongside the Cartesian speed.

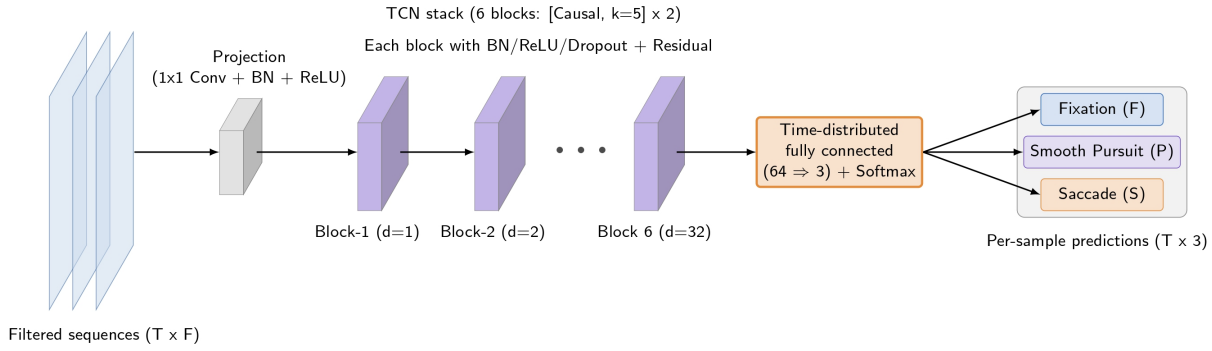


Figure 3.7: TCN diagram: Causal TCN for filtered sequences: dilations 1-32 (kernel 5) provide approximately a 1 s receptive field, with per-sample causal outputs for fixation, smooth pursuit, and saccade, and a fixation minimum-duration constraint.

Network Architecture

The model is implemented using PyTorch, and the overall model design is illustrated in Figure 3.7. The TCN consists of an initial feature projection followed by stacked temporal convolutional blocks with progressively increasing dilation:

- Initial convolution: A 1D convolution with kernel size 1 maps the input features to 64 channels, followed by batch normalization and ReLU activation.
- TCN blocks: Six residual blocks, each containing two layers of causal dilated convolutions with kernel size 5 and dilations of 1, 2, 4, 8, 16, and 32. Batch normalization, ReLU activation, and dropout (0.2) are applied within each block.
- Fixation-aware training: A duration constraint penalizes fixation runs shorter than 60 ms (30 samples at 500 Hz), reducing label interleaving.
- Output layer: A fully connected linear layer maps the 64-dimensional representation at each time step to 3 class logits.

With this architecture, the receptive field covers up to 505 previous samples (around 1.01 s at 500 Hz), enabling the network to use a substantial amount of temporal context in its predictions.

A lightweight label smoother with class-specific minimum length constraints is applied before computing the event-level metrics. This smoothing is only for event-level reporting, and sample-level metrics are computed directly from the raw per-sample TCN predictions.

Training Configuration

The model is trained using the Adam optimizer with a learning rate of 0.001, kernel size 5, and dropout of 0.2 for 50 epochs with a batch size of 16 sequences. Data are split at the sequence level (80/20 split).

Unlike the multi-scale windowed processing used in previous TCN models such as Elmadjian et al., 2023, the adapted TCN operates directly on per-sample features and can assign a label to each new sample as soon as it is observed. The aim is to provide an online classifier with a simpler feature pipeline while achieving performance that is at least comparable to earlier approaches. In the experiments presented later, the adapted TCN offers some overall improvement, particularly in saccade detection.

3.6 Baselines

The primary INSLR-BiLSTM model and the adapted TCN model, along with four baseline models, were all trained and evaluated on the same preprocessed and filtered gaze data described in Section 3.3.4. The baseline models tested were NSLR-HMM (Schrayer, 2022), online TCN (Elmadjian et al., 2023), offline TCN (Elmadjian et al., 2021), and 1D CNN-BLSTM (Startsev et al., 2019). The first baseline was developed for KINARM data by Schrayer, 2022. The last three baselines were developed on datasets that include a Noise/Blink label, and their original implementations keep and predict that class. In our data, blinks have already been removed, so we train and report only three classes - fixation, smooth pursuit, and saccade. For each baseline, the hyperparameters specified in the original papers were first tested. This was followed by the evaluation of tuned variants on the

dataset. The configuration achieving the best performance is reported. Since the data are sampled at 500 Hz, 1-second windows correspond to 500 time steps, with all context windows and parameters scaled accordingly.

NSLR-HMM. The NSLR-HMM model combines piecewise linear segmentation with a hidden Markov model to classify eye movements. The model uses features including segment duration, Cartesian speed, average angular velocity, temporal distance to the nearest angular velocity peak, and the Fisher-transformed cosine of the angle to the previous segment, with emission probabilities modeled using Gaussian distributions over these features. This model was applied to the dataset with the original architecture and hyperparameters unchanged, as it was also developed for KINARM data.

TCN (2023). The original online TCN model uses four temporal blocks with dilations (1, 2, 4, 8), kernel size 5, and dropout 0.25, processing precomputed multi-scale speed and direction features computed over causal 1-second context windows. For this implementation, the model was adapted to the KINARM dataset by adjusting the kernel size to 3 and dropout to 0.2 based on preliminary tuning.

TCN (2021). The offline TCN model uses non-causal dilated convolutions with dilations (1, 2, 4, 8, 16, 32, 64, 128), kernel size 8, and dropout 0.30, operating on 1-second context windows of speed and direction features. For this dataset, the original architecture and other hyperparameters were kept, with only the window length scaled to 500 samples to match this data's sampling rate.

1D CNN-BLSTM. The 1D CNN-BLSTM model uses three convolutional layers with 32, 16, and 8 filters, followed by a bidirectional LSTM with 16 units and dropout 0.3, trained on ~1-second context windows of relative gaze position together with speed, acceleration, and direction features

at multiple temporal scales. For this dataset, 500-sample windows were used for the context, batch size was reduced, and dropout was adjusted to 0.25.

3.7 Evaluation Protocol

3.7.1 KINARM Dataset

Two main metrics were used to assess model performance: sample-level accuracy and event-level accuracy. For all models, both sample-level and event-level metrics were calculated to provide a comprehensive assessment of performance. For the INSLR-BiLSTM model, segment-level metrics were also computed to check how well the predicted labels matched the true segment labels.

Sample-level accuracy checks, for each time point, whether the predicted label matches the ground truth label. So it evaluates classification accuracy on a timestep-by-timestep basis, comparing predicted labels directly against ground truth labels for each data sample.

Event-level metrics are designed to evaluate classification at the level of whole eye movement events, not just isolated samples. For each continuous event in the ground truth, the predicted label for that event is assigned by majority vote over all predicted labels of the samples within the event. In other words, the most common predicted label within the true event is chosen as the event's predicted class. This method is less sensitive to brief label switches and is more forgiving than metrics based on intersection over union (IoU). The approach is similar to evaluation protocols described by Hoppe and Bulling, 2016, and has been used in recent work by Elmadjian et al., 2023.

Overall accuracy was calculated for all the models. For both sample-level and event-level metrics, macro F1 score, weighted F1 score, and individual F1 scores for each class were calculated. For the

INSLR-BiLSTM model, confusion matrices were generated for all three evaluation levels (segment, sample, and event-level) in both count and percentage formats.

No class balancing or up-weighting was applied to the training or test data for any model, as this would have created a biased result. The reported performance therefore reflects the abilities of the models to work with the natural distribution of eye movement classes found in the dataset.

3.7.2 GazeCom Dataset

As an exploratory experiment, the INSLR-BiLSTM model was also evaluated on the GazeCom dataset (Dorr et al., 2010) to assess its applicability to 2D fixed-depth eye tracking data. Since the GazeCom dataset includes eye movement labels beyond fixations, saccades, and smooth pursuits, only the sequence portions containing these three classes were extracted. The Leave-One-Video-Out (LOVO) cross-validation procedure used by Startsev et al., 2019 and Elmadjian et al., 2021 was used. In this procedure, each step of training is done with 17 video clips and the model is evaluated on the remaining video clip. This procedure makes sense here, since the GazeCom dataset has a comparatively larger amount of labeled data.

For this dataset, sample-level and event-level metrics were calculated for the INSLR-BiLSTM model, including overall accuracy, macro F1 score, weighted F1 score, and individual F1 scores for each class. This experiment serves as a proof-of-concept that the model can be adapted to fixed-depth gaze data.

Chapter 4

Results and Discussion

This section presents the evaluation results of the INSLR-BiLSTM model, the Adapted TCN model, and the baseline models on the KINARM dataset. The INSLR-BiLSTM model achieved an overall accuracy of 88% with a macro F1 score of 87.7% and weighted F1 of 90.1%. As shown in Table 4.1, the INSLR-BiLSTM model outperformed all baseline models, with the next best performing model (NSLR-HMM) achieving an accuracy 3.7% lower. These results demonstrate the effectiveness of the INSLR-BiLSTM approach for eye movement classification on pursuit-rich, depth-varying data.

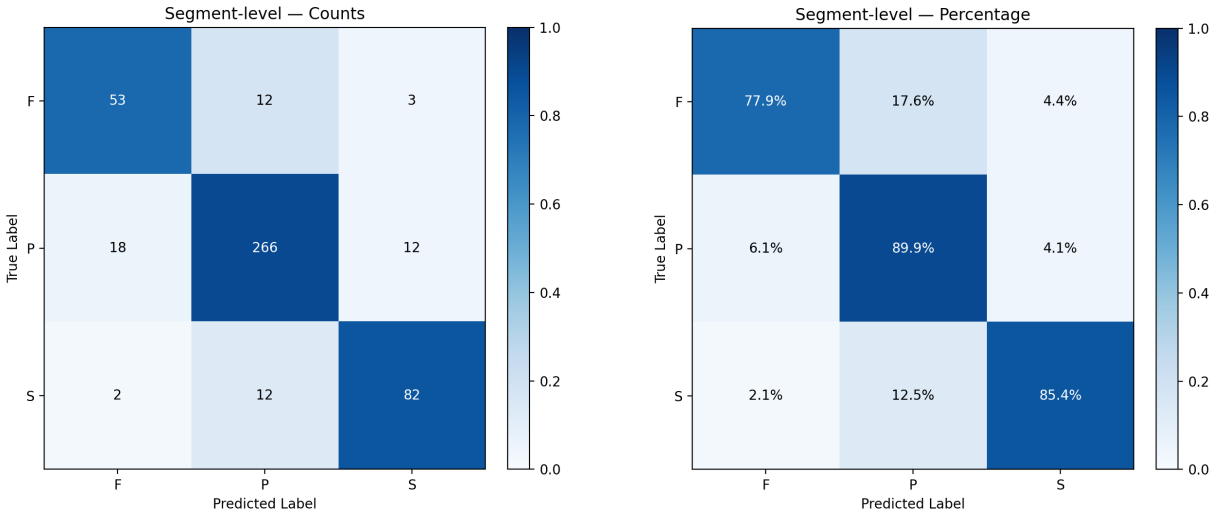
In the remainder of this chapter, the adapted TCN is mainly compared to the TCN (2023) model, since these are the only two online models in the set. The offline models (INSLR-BiLSTM, NSLR-HMM, TCN (2021), and 1D CNN-BLSTM) can use future samples when predicting the current time step. This gives them an advantage over online models in terms of accuracy. At the sample level, the adapted TCN achieves an overall accuracy of 81.1%, which is comparable to the 81.6% obtained by TCN (2023). In the upcoming F1 score results, the adapted TCN shows stronger performance than TCN (2023), particularly for saccade detection.

Table 4.1: Sample-level evaluation results as overall accuracy

Model	Overall Accuracy
INSLR-BiLSTM	**0.880**
Adapted TCN	0.811
NSLR-HMM	0.843
TCN (2023)	0.816
TCN (2021)	0.810
1D CNN-BLSTM	0.785

4.1 Segment-level Results

Segment-level classification accuracy for the INSLR-BiLSTM model was evaluated to verify proper label assignment during the segmentation process. The confusion matrices in Figure 4.1 show that the vast majority of segments were classified correctly, with 89.9% of smooth pursuit segments and 85.4% of saccade segments receiving the correct label. The primary source of confusion was between fixations and smooth pursuits, with 17.6% of fixation segments being misclassified as smooth pursuits (77.9% accurate). This is most likely due to noisy fixations where the gaze signal contains noticeable drift, tremor, or microsaccades. Even after denoising and piecewise linear segmentation, very noisy fixations can still be fragmented into shorter segments. These segments can share kinematic properties with smooth pursuits, making them difficult to distinguish at the segment level.



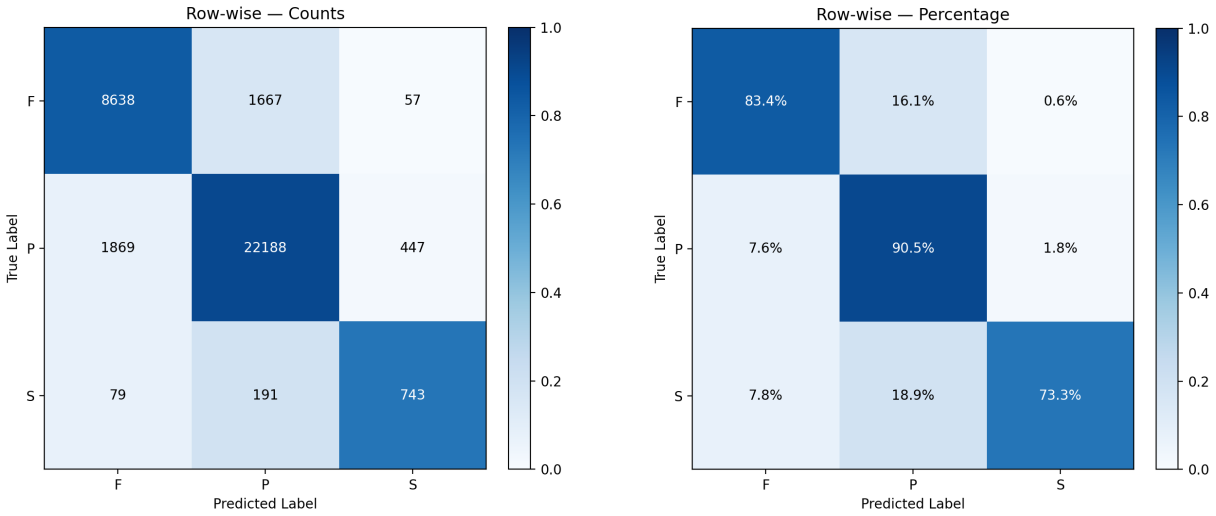
(a) Number-wise confusion matrix

(b) Percentage confusion matrix

Figure 4.1: INSLR-BiLSTM segment-level confusion matrices (Fixation (F), Smooth Pursuit (P), Saccade (S))

4.2 Sample-level Results

Sample-level metrics evaluate classification accuracy on a timestep-by-timestep basis across all data samples. The confusion matrices for the INSLR-BiLSTM model can be seen in Figure 4.2. The INSLR-BiLSTM model achieved better performance across classes compared to other models, with high F1 scores for fixations (0.825) and smooth pursuits (0.914), while saccade detection was more challenging with an F1 score of 0.658 (Table 4.2). This pattern is also visible in the confusion matrices, where most samples lie on the main diagonal and the largest off-diagonal errors involve saccades. The reason for the lower saccade F1 score compared to the other eye movements is explored after event-level results are presented.



(a) Number-wise confusion matrix

(b) Percentage confusion matrix

Figure 4.2: INSLR-BiLSTM sample (row)-level confusion matrices (Fixation (F), Smooth Pursuit (P), Saccade (S))

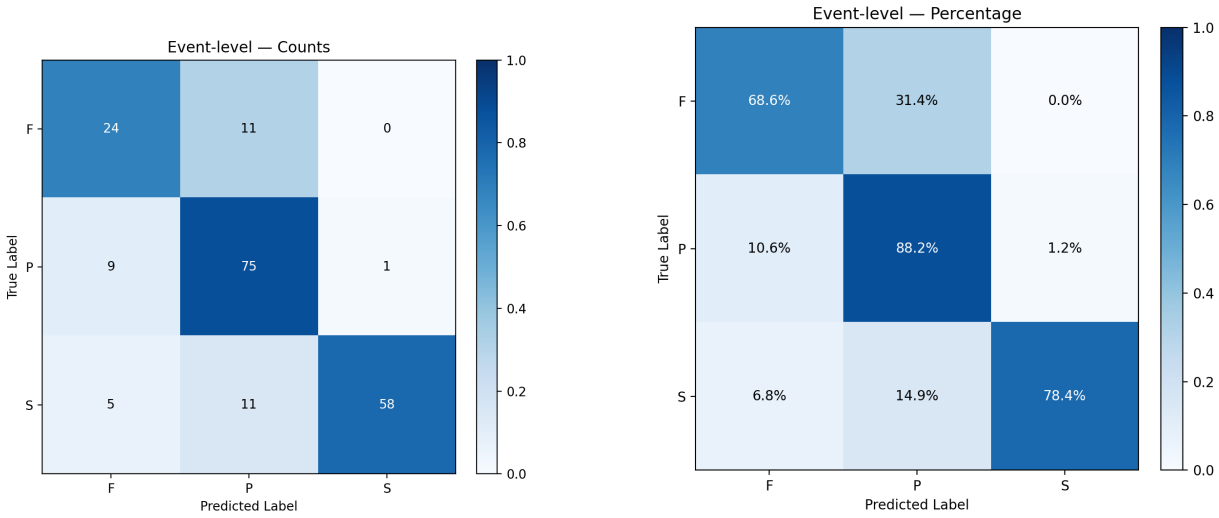
Table 4.2: Sample-level evaluation results as F1 scores

Model	Macro F1	Weighted F1	Fixation	Saccade	Smooth Pursuit
INSLR-BiLSTM	**0.877**	**0.901**	**0.825**	**0.658**	**0.914**
Adapted TCN	0.740	0.811	0.763	0.610	0.847
NSLR-HMM	0.762	0.847	0.809	0.600	0.876
TCN (2023)	0.699	0.811	0.736	0.504	0.858
TCN (2021)	0.726	0.826	0.755	0.555	0.870
1D CNN-BLSTM	0.614	0.796	0.730	0.266	0.847

Comparing across all models, the INSLR-BiLSTM model substantially outperforms the baselines in macro F1 (0.877 vs. 0.614-0.762), weighted F1 (0.901 vs. 0.796-0.847), and in individual class F1 scores. NSLR-HMM is the next best model, with a macro F1 of 0.762 and a weighted F1 of 0.847. The 1D CNN-BLSTM baseline performed notably poorly on saccade detection (F1 score of 0.266), suggesting that saccade classification is challenging for some architectures. The adapted TCN model performed well for an online model, achieving a macro F1 of 0.740 compared to the TCN (2023) model's 0.699. The results for the adapted TCN show improved classification, especially for saccades (0.610 vs. 0.504 F1). Across all models, smooth pursuits consistently achieve the highest F1 scores, fixations perform moderately, and saccades are the most difficult class to detect at the sample level.

4.3 Event-level Results

Event-level metrics evaluate performance at the level of complete eye movement events. The confusion matrices for the INSLR-BiLSTM model can be seen in Figure 4.3. At this level, the INSLR-BiLSTM model achieved an event-level accuracy of 0.809, with a macro F1 of 0.785 and weighted F1 of 0.812 (Table 4.3). Overall, the INSLR-BiLSTM model outperformed all baselines, with the largest gap observed in macro F1 scores (0.785 vs. 0.530-0.718 for other models). At the event level, saccade detection improved to an F1 of 0.872, while fixation detection was only comparable to the other models. Since fixations had a sample-level F1 score of 0.825, an event-level F1 score of 0.658 indicates that some fixation segments were misclassified as smooth pursuits. Because event-level scoring treats each ground-truth fixation as a single unit, misclassified segments



(a) Number-wise confusion matrix

(b) Percentage confusion matrix

Figure 4.3: INSLR-BiLSTM event-level confusion matrices (Fixation (F), Smooth Pursuit (P), Saccade (S))

inside that fixation can cause the entire event to be counted as wrong. This pulls the event-level F1 down relative to the sample-level F1. In practice, if several segments within a fixation are labeled as smooth pursuit, the majority vote over that event can flip the entire event label from fixation to smooth pursuit. This happens with noisy fixations or ambiguous boundaries between fixations and pursuits.

TCN (2021) achieved slightly higher fixation F1 (0.673) than INSLR-BiLSTM but lower scores for saccades and smooth pursuits, resulting in a lower macro F1 overall. At the event level, the adapted TCN maintained its advantage over the original TCN (2023) model with a macro F1 of 0.718 versus 0.657, and higher F1 scores for both saccades (0.779 vs. 0.625) and smooth pursuits (0.716 vs. 0.677). These results show that the adapted TCN gives consistently better event-level performance than the original online TCN on this dataset.

Table 4.3: Event-level evaluation results as overall event-level accuracy and F1 scores

Model	Accuracy	Macro F1	Weighted F1	Fixation	Saccade	Smooth Pursuit
INSLR-BiLSTM	**0.809**	**0.785**	**0.812**	0.658	**0.872**	**0.824**
Adapted TCN	0.717	0.718	0.722	0.660	0.779	0.716
NSLR-HMM	0.716	0.696	0.724	0.585	0.762	0.742
TCN (2023)	0.661	0.657	0.656	0.670	0.625	0.677
TCN (2021)	0.729	0.722	0.729	**0.673**	0.745	0.747
1D CNN-BLSTM	0.569	0.530	0.519	0.638	0.308	0.643

Although the sample-level saccade F1 scores for the INSLR-BiLSTM model were higher than those for other models, it was only 0.658. However, the event-level F1 score for saccades was 0.872. To understand this further, the sample-level metrics were recomputed with tolerance in the event boundaries. For this, a tolerance of 20 samples was used for fixations and smooth pursuits, and a tolerance of 5 samples was used for saccades. Tolerant sample-level evaluation is the sample-level accuracy after allowing a small boundary tolerance: a row counts as correct if the model predicts the ground-truth class anywhere within a local window around that row. The sample-level results using tolerance are displayed in Table 4.4. The F1 scores of fixations and smooth pursuits did not change significantly. However, the score for saccades jumped from 0.658 to 0.863. This, along with the event-level score of 0.872, indicates that the saccades have been localized properly. But the exact event boundaries of the saccades are shifted by a few samples relative to the ground truth. Since saccades are very short, even a boundary error of a few rows can substantially decrease the sample-level accuracy. The boundary error could be caused by the value used for the INSLR algorithm's clustering frequency, which is 20 samples. This value was chosen since it gave the

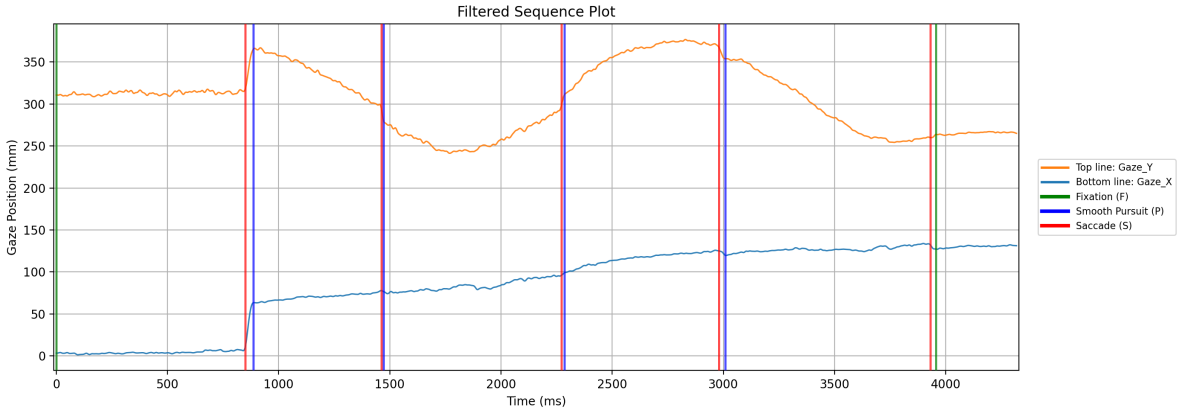
best trade-off between over-fragmentation of eye movements, and proper identification of shorter saccades. However, the average duration of saccades in the dataset is 30 ms which corresponds to 15 samples (at 500 Hz). Since the value of the clustering frequency is higher than the average saccade duration, this is most likely causing the imprecise boundary placement for some of the saccades, because candidate boundary splits would only be considered every 20 samples.

Table 4.4: Tolerant sample-level evaluation results as F1 scores

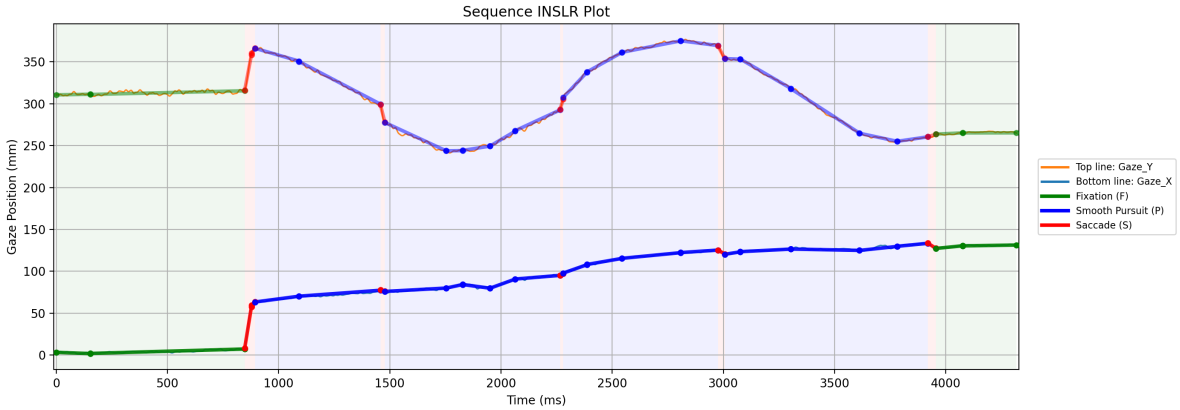
Model	Macro F1	Weighted F1	Fixation	Saccade	Smooth Pursuit
INSLR-BiLSTM	0.877	0.901	0.839	0.863	0.928

4.4 Results Visualization

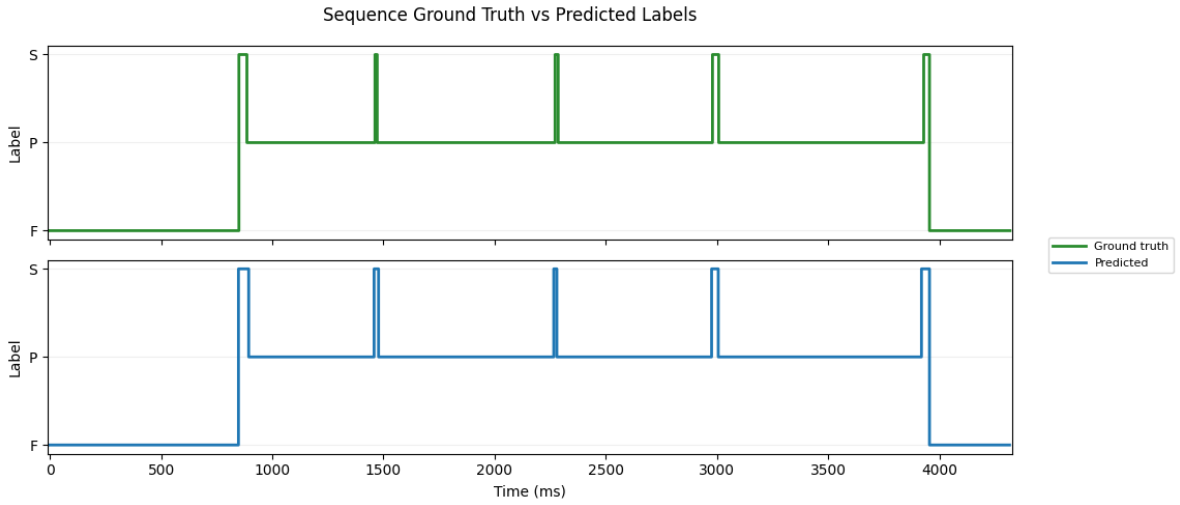
Figure 4.4 shows an example of successful prediction, where the predicted labels closely match the ground-truth labels across most of the sequence (as shown in the third subplot, Figure 4.4c). The saccades in particular are localized correctly. However, examining this example more closely reveals that while the model identifies saccade locations well, the saccade boundaries are not always precise, as can be seen in the small differences between the widths of the predicted and ground-truth label bands.



(a)



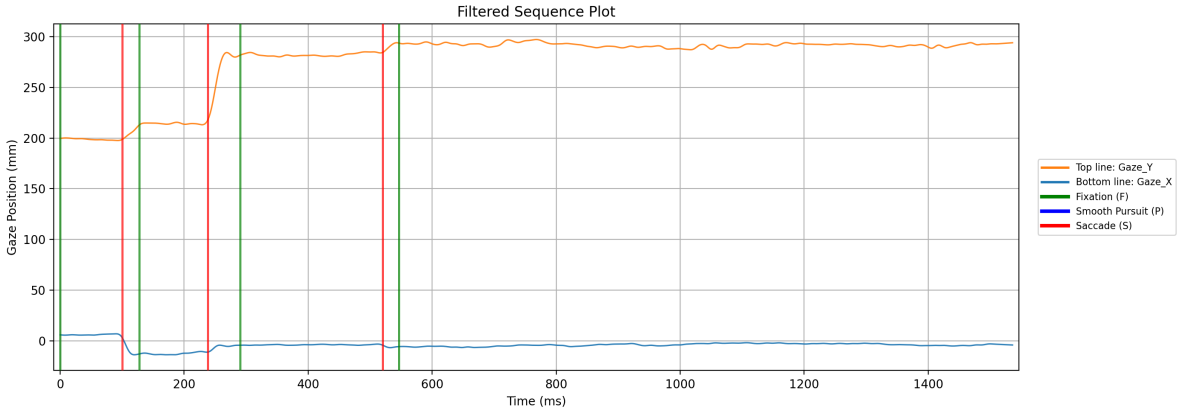
(b)



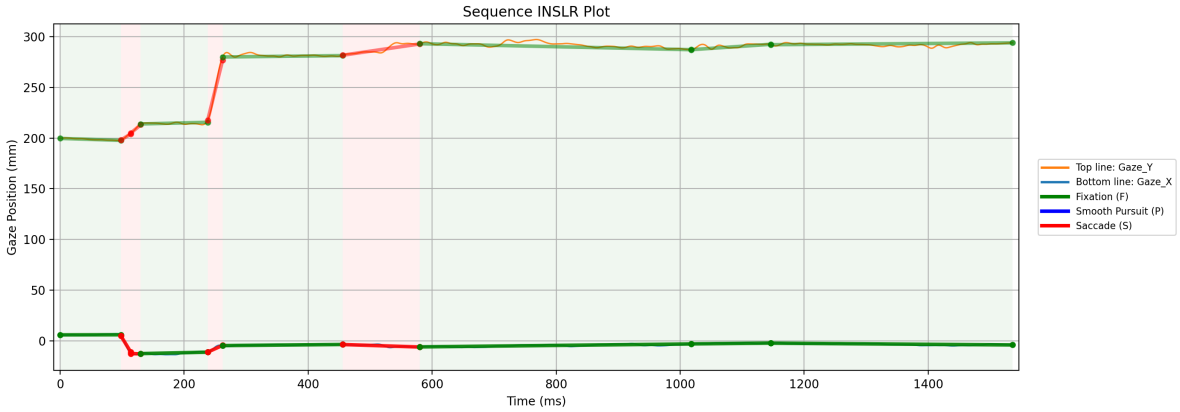
(c)

Figure 4.4: (a) Filtered input. (b) INSLR segments. (c) Ground-truth vs model predictions over time (Fixation (F), Smooth Pursuit (P), Saccade (S)). Illustrates a well-classified sequence.

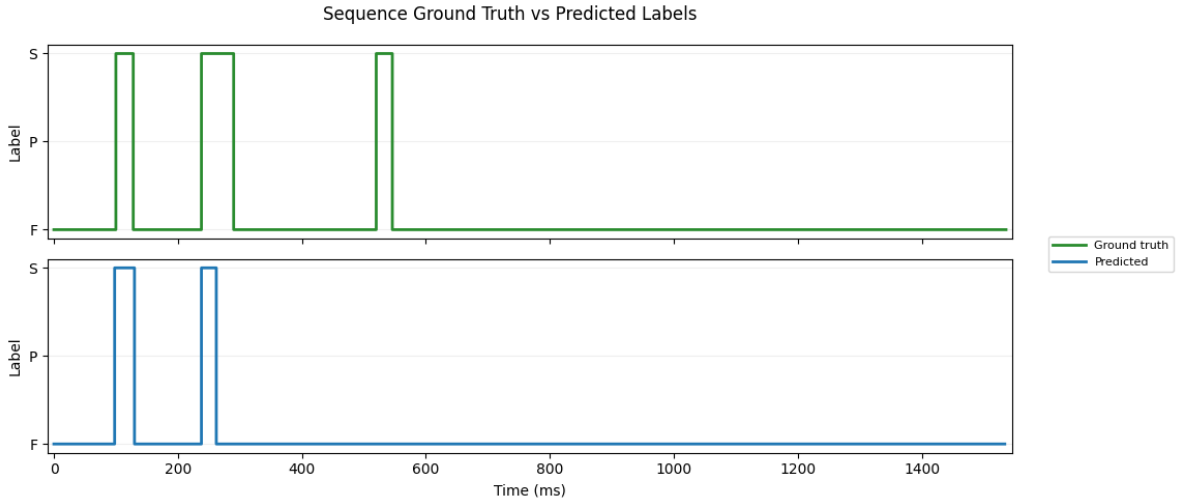
Figure 4.5 shows this boundary error more clearly. For the second saccade, the point where the predicted label ends is slightly offset from the true event boundary. This pattern explains the gap between the sample-level and event-level accuracy values for saccades. The model successfully detects that a saccade is present and roughly where it occurs, but the exact boundary is not captured precisely. The figure also shows a saccade that the model does not identify, which helps explain why the event-level F1 score for saccades does not rise above 0.9.



(a)



(b)



(c)

Figure 4.5: (a) Filtered input. (b) INSLR segments. (c) Ground-truth vs model predictions over time for a different sequence (Fixation (F), Smooth Pursuit (P), Saccade (S)). Illustrates boundary error for saccades, and a saccade that has not been identified.

4.5 GazeCom Results

As a preliminary generalization test, the INSLR-BiLSTM model was evaluated on the GazeCom dataset and achieved good results at both the sample and event levels. Table 4.5 shows that sample-level accuracy reached 90.7%, with a macro F1 of 0.827 and weighted F1 of 0.903. At the event level, accuracy was 91.3%, with macro F1 of 0.846 and weighted F1 of 0.913. These results are broadly comparable to those reported in eye movement detection studies in recent literature. This suggests that the model can also be applied to 2D fixed-depth gaze data, despite being developed for depth-varying data in a robotic workspace. This preliminary experiment indicates that the model may be able to generalize across different eye tracking setups.

Table 4.5: GazeCom evaluation results as F1 scores

Level	Accuracy	Macro F1	Weighted F1	Fixation	Saccade	Smooth Pursuit
Sample-level	0.907	0.827	0.903	0.944	0.809	0.729
Event-level	0.913	0.846	0.913	0.920	0.933	0.684

Chapter 5

Conclusion

5.1 Conclusions

This work introduced two models for automated eye movement classification in a depth-varying robotic workspace. These are the INSLR-BiLSTM model and an adapted online TCN model. The INSLR-BiLSTM model was designed as an offline, segment-based classifier tailored to the KINARM dataset. Because most existing deep learning approaches were developed for fixed-depth, screen-based data and did not perform as well on the KINARM recordings, a different strategy was needed. In this work, a two-stage pipeline was used that first segments the gaze signal and then classifies the resulting segments. By combining the INSLR segmentation method with a BiLSTM architecture, the model achieved 88.0% overall sample-level accuracy on the KINARM dataset. It outperformed all the baseline models in this study. The results show that this method is effective for classifying fixations, saccades, and smooth pursuits in robotic depth-varying environments.

However, the model still requires improvement in two main areas. These are precisely localizing saccade boundaries and distinguishing parts of fixations from smooth pursuits. These limitations

can be seen in the gap between sample-level and event-level metrics for saccades and fixations. Still, the model provides a solid foundation for further exploration into automated eye movement analysis for different types of data. The INSLR-BiLSTM model showed reasonable generalization to fixed-depth eye-tracking data from the GazeCom dataset, with good performance at both sample and event levels. This suggests that it may be used with other types of gaze data as well.

The adapted TCN model was developed as a causal, per-sample alternative that can run online and produce labels in real time. The original online TCN by Elmadjian et al., 2023 relies on multi-scale features computed over causal context windows. On the other hand, the adapted TCN works directly with per-sample features and includes a minimum fixation-duration constraint. On the KINARM dataset, the adapted TCN achieved sample-level accuracy comparable to the TCN (2023) baseline. It also produced higher macro and weighted F1 scores and better saccade detection at both sample and event levels. This suggests that an online, fixation-aware TCN with a simpler feature pipeline can still perform at least as well as earlier online models on this task.

Tables 5.1 and 5.2 summarize the intended use, strengths, limitations, and key metrics for both models.

Table 5.1: Summary of the INSLR-BiLSTM model

Model	INSLR-BiLSTM
Setting and intended use	• Offline analysis on saved recordings • Analysis tailored to KINARM data in a depth-varying workspace • Best for pursuit-rich sequences
Strengths	• Highest overall performance in this study • Segment first design reduces label flicker • Strong smooth pursuit detection • Interpretable segment-level features
Known limitations	• Saccade onset and offset may be imprecise, although localization is good • Can be affected by movements within noisy fixations • Not intended for real-time use
Key metrics (this study)	• Sample-level accuracy 88.0% on KINARM • Event-level macro F1 0.785 overall • Smooth pursuit sample-level F1 0.914

Table 5.2: Summary of the adapted TCN model

Model	Adapted TCN
Setting and intended use	• Online, causal, per sample classification • Can be used for real-time labeling of KINARM data in a depth-varying workspace • Receptive field of about 1s of past context
Strengths	• Per sample outputs enable immediate response • Fixation minimum duration reduces label interleaving • Better overall performance and saccade detection than TCN (2023) in this study
Known limitations	• Overall accuracy lower than some of the offline models • Less interpretable than the segment based pipeline
Key metrics (this study)	• Sample-level accuracy 81.1% on KINARM • Event-level macro F1 0.718 overall • Saccade detection advantage over TCN (2023) baseline

5.2 Limitations

5.2.1 Dataset and Generalization

The model was developed and evaluated on the KINARM dataset, which contains depth-varying gaze data collected in a robotic workspace under controlled laboratory conditions (for example, with a fixed head position). Hence, its generalization to other eye-tracking data or less constrained settings remains to be fully explored. The evaluation on the GazeCom dataset was only a preliminary check and was carried out only for the INSLR-BiLSTM model without comparison to other methods. This limits the conclusions we can make about broader applicability.

5.2.2 Classification Challenges

The INSLR-BiLSTM model still has two main classification difficulties. First, at the segment level, around 17.6% of fixation segments are misclassified as smooth pursuits. This is also reflected in the lower event-level F1 score for fixations compared to the other classes. This often occurs for noisy fixations that are segmented into multiple unstable segments instead of a few stable ones. The segments that contain movements such as microsaccades, drift, or tremor can have a low but non-zero velocity profile that looks similar to slow pursuit.

Second, saccades are usually detected in the right place, but their start and end boundaries are not always aligned with the ground-truth events. This leads to a lower sample-level F1 score for saccades, even though the event-level and tolerant sample-level metrics are higher. Very short

saccades are especially affected, because even small boundary errors or a missed event can reduce the F1 score.

5.2.3 Label Quality and Training Data

Although the manual labels were carefully reviewed, they may still contain occasional errors or ambiguous boundaries. In addition, only four participants had labeled data in this study. This limits both the amount of training data and the range of individual differences seen by the models. With more labeled participants, it would be possible to train on $n - 1$ participants and test on the held-out participant, giving a clearer picture of how well the models generalize across people. A larger labeled dataset could also make the models more robust overall. Even with these limitations, the trained model can already be used to automatically label new sequences. Human annotators can use this to help them manually label more data.

5.3 Future Work

Future directions for this work include testing the model on additional eye-tracking datasets and on experimental setups other than the depth-varying robotic workspace used here. Testing the approach on other time-series classification problems would help assess whether this segmentation-based BiLSTM framework is applicable beyond eye movement detection. A key priority is to investigate the gap between sample-level and event-level performance for fixations and saccades. This involves trying alternative feature sets and examining why the gap arises. Another direction is to evaluate feature fusion to determine whether jointly using features improves robustness near boundaries and

under noise. The models could also be extended to predict blinks and noisy intervals, which would make it more suitable for real-world eye-tracking data. With enough labeled data, the models could be further extended to detect smooth pursuits and saccades, with or without a vergence component. This would help separate eye movements that involve depth changes. Such extensions would help establish the generalizability and improve the accuracy of the models presented in this work.

References

Agtzidis, I., Startsev, M., & Dorr, M. (2016). In the pursuit of (ground) truth: A hand-labelling tool for eye movements recorded during dynamic scene viewing. *2016 IEEE Second Workshop on Eye Tracking and Visualization (ETVIS)*, 65–68.

Al-Adhaileh, M. H., Alsubari, S. N. M., Al-Nefaie, A. H., Ahmad, S., & Alhamadi, A. A. (2025). Diagnosing autism spectrum disorder based on eye tracking technology using deep learning models. *Frontiers in Medicine*, *Volume 12 - 2025*.

Andersson, R., Larsson, L., Holmqvist, K., Stridh, M., & Nyström, M. (2017). One algorithm to rule them all? an evaluation and discussion of ten eye movement event-detection algorithms. *Behavior Research Methods*, *49*(2), 616–637.

Barany, D. A., Gómez-Granados, A., Schrayer, M., Cutts, S. A., & Singh, T. (2020). Perceptual decisions about object shape bias visuomotor coordination during rapid interception movements. *Journal of Neurophysiology*, *123*(6), 2235–2248.

Birawo, B., & Kasprowski, P. (2022). Review and evaluation of eye movement event detection algorithms. *Sensors (Basel, Switzerland)*, *22*(22).

Butterworth, S., et al. (1930). On the theory of filter amplifiers. *Wireless Engineer*, *7*(6), 536–541.

Chauhan, M., Thorwe, P., Mukherjee, M. J., & Rao, Y. S. (2018). Sensor data analysis using moving average filter and 256-point FFT for wireless sensor networks. *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 1–7.

Cheng, K. Y., Rehani, M., & Hebert, J. S. (2023). A scoping review of eye tracking metrics used to assess visuomotor behaviours of upper limb prosthesis users. *Journal of NeuroEngineering and Rehabilitation*, *20*(1), 49.

Costela, F. M., McCamy, M. B., Macknik, S. L., Otero-Millan, J., & Martinez-Conde, S. (2013). Microsaccades restore the visibility of minute foveal targets. *PeerJ*, *1*, e119.

Culicetto, L., Cardile, D., Marafioti, G., Lo Buono, V., Ferraioli, F., Massimino, S., Di Lorenzo, G., Sorbera, C., Brigandì, A., Vicario, C. M., Quartarone, A., & Marino, S. (2025). Recent advances (2022-2024) in eye-tracking for parkinson's disease: A promising tool for diagnosing and monitoring symptoms. *Frontiers in aging neuroscience*, *17*, 1534073.

Dar, A. H., Wagner, A. S., & Hanke, M. (2021). REMoDNaV: Robust eye-movement classification for dynamic stimulation. *Behavior research methods*, *53*(1), 399–414.

de Brouwer, S., Yuksel, D., Blohm, G., Missal, M., & Lefèvre, P. (2002). What triggers catch-up saccades during visual tracking? *Journal of Neurophysiology*, *87*(3), 1646–1650.

Del Punta, J. A., Rodriguez, K. V., Gasaneo, G., & Bouzat, S. (2019). Models for saccadic motion and postsaccadic oscillations. *Physical review. E*, *99*(3), 032422.

Dorr, M., Martinetz, T., Gegenfurtner, K. R., & Barth, E. (2010). Variability of eye movements when viewing dynamic natural scenes. *Journal of Vision*, *10*(10), 28–28.

Elmadjian, C., Gonzales, C., Costa, R. L. d., & Morimoto, C. H. (2023). Online eye-movement classification with temporal convolutional networks. *Behavior research methods*, *55*(7), 3602–3620.

Elmadjian, C., Gonzales, C., & Morimoto, C. H. (2021). Eye movement classification with temporal convolutional networks. *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part III*, 390–404.

Engbert, R. (2006). Microsaccades: A microcosm for research on oculomotor control, attention, and visual perception. In S. Martinez-Conde, S. L. Macknik, L. M. Martinez, J.-M. Alonso, & P. U. Tse (Eds.), *Visual perception* (pp. 177–192, Vol. 154). Elsevier.

Glaser, J. I., Wood, D. K., Lawlor, P. N., Segraves, M. A., & Kording, K. P. (2020). From prior information to saccade selection: Evolution of frontal eye field activity during natural scene search. *Cerebral cortex (New York, N.Y. : 1991)*, *30*(3), 1957–1973.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5), 602–610.

Guadron, L., Titchener, S. A., Abbott, C. J., Ayton, L. N., van Opstal, A. J., Petoe, M. A., & Goossens, J. (2024). Post-saccadic oscillations of the pupil and lens reduce fixation stability in retinitis pigmentosa and age-related macular degeneration. *Investigative ophthalmology & visual science*, *65*(5), 39.

Guestrin, E., & Eizenman, M. (2006). General theory of remote gaze estimation using the pupil center and corneal reflections. *IEEE Transactions on Biomedical Engineering*, *53*(6), 1124–1133.

Harezlak, K., & Kasprowski, P. (2019). Understanding eye movement signal characteristics based on their dynamical and fractal features. *Sensors*, *19*(3).

Heinen, S. J., Potapchuk, E., & Watamaniuk, S. N. J. (2016). A foveal target increases catch-up saccade frequency during smooth pursuit. *Journal of Neurophysiology*, *115*(3), 1220–1227.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780.

Hoppe, S., & Bulling, A. (2016). End-to-end eye movement detection using convolutional neural networks. *CoRR*, *abs/1609.02452*.

Jameer, S., & Syed, H. (2023). Deep SE-BiLSTM with IFPOA fine-tuning for human activity recognition using mobile and wearable sensors. *Sensors*, *23*(9).

Johari, K., Bhardwaj, R., Kim, J.-J., Yow, W. Q., & Tan, U.-X. (2024). Eye movement analysis for real-world settings using segmented linear regression. *Computers in biology and medicine*, *174*, 108364.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45.

Khatun, M. A., Yousuf, M. A., Ahmed, S., Uddin, M. Z., Alyami, S. A., Al-Ashhab, S., Akhdar, H. F., Khan, A., Azad, A., & Moni, M. A. (2022). Deep CNN-LSTM with self-attention model for human activity recognition using wearable sensor. *IEEE Journal of Translational Engineering in Health and Medicine*, *10*, 1–16.

Klaib, A. F., Alsrehin, N. O., Melhem, W. Y., Bashtawi, H. O., & Magableh, A. A. (2021). Eye tracking algorithms, techniques, tools, and applications with an emphasis on machine learning and internet of things technologies. *Expert Systems with Applications*, *166*, 114037.

Ko, H.-K., Snodderly, D. M., & Poletti, M. (2016). Eye movements between saccades: Measuring ocular drift and tremor. *Vision research*, *122*, 93–104.

Komogortsev, O. V., & Karpov, A. (2013). Automated classification and scoring of smooth pursuit eye movements in the presence of fixations and saccades. *Behavior Research Methods*, *45*(1), 203–215.

Krauzlis, R. J., Goffart, L., & Hafed, Z. M. (2017). Neuronal control of fixation and fixational eye movements. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, *372*(1718).

Mahanama, B., Jayawardana, Y., Rengarajan, S., Jayawardena, G., Chukoskie, L., Snider, J., & Jayarathna, S. (2022). Eye movement and pupil measures: A review. *Frontiers in Computer Science*, *Volume 3 - 2021*.

Martinez-Conde, S., Macknik, S. L., Troncoso, X. G., & Dyar, T. A. (2006). Microsaccades counteract visual fading during fixation. *Neuron*, *49*(2), 297–305.

Mejia-Romero, S., Eduardo Lugo, J., Bernardin, D., & Faubert, J. (2021). An effective filtering process for the noise suppression in eye movement signals. In K. Ray, K. C. Roy, S. K. Toshniwal, H. Sharma, & A. Bandyopadhyay (Eds.), *Proceedings of international conference on data science and applications* (pp. 33–46). Springer Singapore.

Mestre, C., Gautier, J., & Pujol, J. (2018). Robust eye tracking based on multiple corneal reflections for clinical applications. *Journal of biomedical optics*, *23*(3), 1–9.

Nachmani, O., Coutinho, J., Khan, A. Z., Lefèvre, P., & Blohm, G. (2020). Predicted position error triggers catch-up saccades during sustained smooth pursuit. *eNeuro*, *7*(1), ENEURO.0196–18.2019.

Nyström, M., Andersson, R., Niehorster, D. C., Hessels, R. S., & Hooge, I. T. C. (2024). What is a blink? classifying and characterizing blinks in eye openness signals. *Behavior Research Methods*, *56*(4), 3280–3299.

Pavisic, I. M., Firth, N. C., Parsons, S., Rego, D. M., Shakespeare, T. J., Yong, K. X. X., Slattery, C. F., Paterson, R. W., Foulkes, A. J. M., Macpherson, K., Carton, A. M., Alexander, D. C., Shawe-Taylor, J., Fox, N. C., Schott, J. M., Crutch, S. J., & Primativo, S. (2017). Eyetracking metrics in young onset alzheimer's disease: A window into cognitive visual functions. *Frontiers in Neurology*, *Volume 8 - 2017*.

Pekkanen, J., & Lappi, O. (2017). A new and general approach to signal denoising and eye movement classification based on segmented linear regression. *Scientific reports*, *7*(1), 17726.

Purves, D., Augustine, G. J., Fitzpatrick, D., Katz, L. C., LaMantia, A.-S., McNamara, J. O., & Williams, S. M. (2001). Types of eye movements and their functions. In *Neuroscience. 2nd edition*. Sinauer Associates.

Qin, H., Zhu, H., Jin, X., Song, Q., El-Yacoubi, M. A., & Gao, X. (2024). EmMixformer: Mix transformer for eye movement recognition.

Raju, M. H., Friedman, L., Bouman, T. M., & Komogortsev, O. V. (2021). Filtering eye-tracking data from an EyeLink 1000: Comparing heuristic, savitzky-golay, IIR and FIR digital filters. *Journal of Eye Movement Research*, *14*(3), 1–16.

Rogers, S. L., Speelman, C. P., Guidetti, O., & Longmuir, M. (2018). Using dual eye tracking to uncover personal gaze patterns during social interaction. *Scientific Reports*, *8*(1), 4271.

Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols [event-place: Palm Beach Gardens, Florida, USA]. *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, 71–78.

Santini, T., Fuhl, W., Kübler, T., & Kasneci, E. (2016). Bayesian identification of fixations, saccades, and smooth pursuits [event-place: Charleston, South Carolina]. *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, 163–170.

Savitzky, A., & Golay, M. J. E. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, *36*(8), 1627–1639.

Schrayer, M. F. (2022). *Automating the segmentation of eye movements in a robotic virtual reality environment* [Master's thesis]. University of Georgia.

Schütz, A. C., Kerzel, D., & Souto, D. (2014). Saccadic adaptation induced by a perceptual task. *Journal of Vision*, *14*(5), 4–4.

Singh, T., Perry, C. M., & Herter, T. M. (2016). A geometric method for computing ocular kinematics and classifying gaze events using monocular remote eye tracking in a robotic environment. *Journal of neuroengineering and rehabilitation*, *13*, 10.

Špakov, O. (2012). Comparison of eye movement filters used in HCI [event-place: Santa Barbara, California]. *Proceedings of the Symposium on Eye Tracking Research and Applications*, 281–284.

Startsev, M., Agtzidis, I., & Dorr, M. (2019). 1d CNN with BLSTM for automated classification of fixations, saccades, and smooth pursuits. *Behavior research methods*, *51*(2), 556–572.

Suhari, A. R., Hartanto, R., & Wibirama, S. (2023). Improving performance of eye movements classification using CNN-transformer model. *2023 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, 751–756.

Tokushige, S.-i., Matsumoto, H., Matsuda, S.-i., Inomata-Terada, S., Kotsuki, N., Hamada, M., Tsuji, S., Ugawa, Y., & Terao, Y. (2023). Early detection of cognitive decline in alzheimer's disease using eye tracking. *Frontiers in Aging Neuroscience*, *Volume 15 - 2023*.

Trabulsi, J., Norouzi, K., Suurmets, S., Storm, M., & Ramsøy, T. Z. (2021). Optimizing fixation filters for eye-tracking on small screens. *Frontiers in Neuroscience*, *Volume 15 - 2021*.

Valliappan, N., Dai, N., Steinberg, E., He, J., Rogers, K., Ramachandran, V., Xu, P., Shojaeizadeh, M., Guo, L., Kohlhoff, K., & Navalpakkam, V. (2020). Accelerating eye movement research via accurate and affordable smartphone eye tracking. *Nature Communications*, *11*(1), 4553.

Wang, C., Wang, R., Leng, Y., Iramina, K., Yang, Y., & Ge, S. (2024). An eye movement classification method based on cascade forest. *IEEE journal of biomedical and health informatics*, *28*(12), 7184–7194.

Wang, C.-M., & Hsu, W.-C. (2024). Design of a gaze-controlled interactive art system for the elderly to enjoy life. *Sensors*, *24*(16).

Warman, R. A., Wibirama, S., & Bejo, A. (2017). Performance comparison of signal processing filters on smooth pursuit eye movements. *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 111–115.

Weickert, J. (1998). *Anisotropic diffusion in image processing*. Teubner.

Xie, J., Chen, R., Liu, Z., Zhou, J., Hou, J., & Zhou, Z. (2025). GMM-HMM-based eye movement classification for efficient and intuitive dynamic human-computer interaction systems. *Journal of eye movement research*, *18*(4), 28.

Zemblys, R., Niehorster, D. C., Komogortsev, O., & Holmqvist, K. (2018). Using machine learning to detect events in eye-tracking data. *Behavior Research Methods*, *50*(1), 160–181.