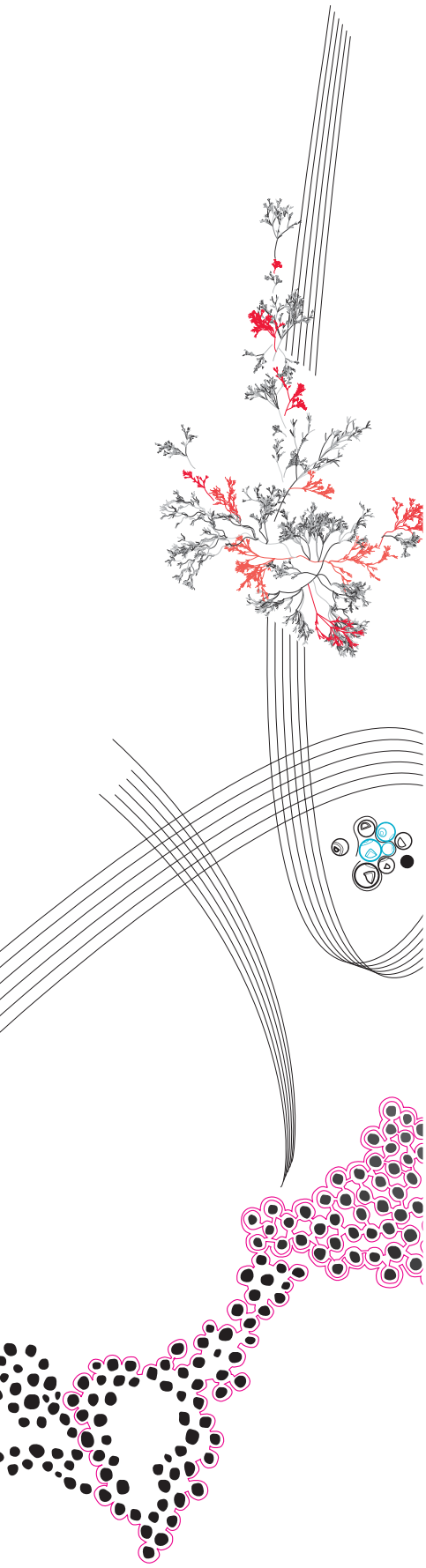


MSc Thesis Embedded Systems

Earable-Based Visual Distraction Monitoring in Cyclists

Sidhharth Balakrishnan

Supervisors:
Dr. O. Durmaz - Incel
A.R. Pallamreddy
Dr.ing. Y. Huang
Dr.Ir. D. Reidsma

June, 2025

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Earable-Based Visual Distraction Monitoring in Cyclists

Sidhharth Balakrishnan
University of Twente
Enschede, the Netherlands
s.balakrishnan@student.utwente.nl

Akhil Pallamreddy
University of Twente
Pervasive System Research Group
Enschede, the Netherlands
a.r.pallamreddy@utwente.nl

Özlem Durmaz Incel
University of Twente
Pervasive System Research Group
Enschede, the Netherlands
ozlem.durmaz@utwente.nl

***Abstract*—Visual distractions among cyclists significantly lower cyclists' situational awareness, heightening the risk of accidents. This paper proposes the utilization of an open-source OpenEarable device, which is equipped with onboard inertial measurement units (IMU), as an easy and non-invasive method for detecting visual distractions through the quantification of head movements that are indicative of behaviours associated with visual distraction. Head movement patterns of 20 subjects were recorded using earable IMU sensors in naturalistic cycling scenarios. Classical machine learning and deep learning models were employed to analyze the collected data and identify patterns characteristic of visual distractions in cyclists. Among machine learning models, Support Vector Machine (SVM) achieved the highest F1-score (85%) with a fair Kappa score (0.59). In deep learning models, Convolutional Neural Network (CNN) offers the best F1-score (87%) and a substantial Kappa score (0.74). To asses edge-deployment feasibility, the models are further optimized and evaluated for deployment on Raspberry Pi. The LinearSVC variant of the SVM model offers the best trade-off between model size, inference time, and classification performance. Whereas, for CNN, quantization techniques like Post Training Quantization (PTQ) and Quantization-Aware Training (QAT) reduce the model size without sacrificing performance. These results highlight the potential of earable devices for real-time distraction detection and provide a foundation for future wearable mobility safety systems.**

***Index Terms*—Earables, Inertial Measurement Unit, Head gestures, Edge computing, Quantization**

I. Introduction

Cycling offers a sustainable mode of transportation that reduces traffic congestion, environmental pollution, and contributes to better physical and mental health [1]. Regular cycling substantially reduces the risk of cardiovascular disease, type 2 diabetes, and also alleviates anxiety and depression [1]. Due to these benefits, cycling continues to gain popularity as both a recreational activity and a means of transportation. However, cyclists are among the most vulnerable road users, primarily due to the fact that they have less physical protection compared to motor vehicle occupants shielded by protective devices like airbags and roll cages [2]. Consequently, cyclists are far more vulnerable to severe or fatal injuries in traffic accidents [3]. Although helmets significantly decrease head injuries [4], the overall physical vulnerability persists.

A growing concern regarding bicycle safety is the rise in visual distractions among cyclists. Visual distractions include observing objects on or near the road and engaging with personal devices, which can significantly impair situational awareness in cyclists [5]. The distractions lower cyclists' situational awareness, decrease reaction time, and increase risky behavior such as lane weaving or violating traffic signals, thereby raising the chances of accidents by a significant margin [2]. In the Netherlands, cyclists were involved in approximately 68% of the road injuries as per the 2020 hospital registry, most frequently as a result of non-motor vehicle collisions, like those resulting from distractions [4], [6]. Surveys in the Netherlands reveal an increase in device (smartphones) interaction among cyclists from 19% in 2015 to 28% in 2019 [2]. Younger cyclists are particularly vulnerable to visual distraction, as they tend to attend to roadside scenery, mobile phones, navigation, and media interaction, which significantly increases their risk of injury [4].

Cycling behavior has been studied with observational studies and questionnaires [1], [7], [8]. While effective in determining general patterns of risk [9], they suffer from recall bias, lack real-time feedback, and cannot capture subtle, dynamic motion—particularly brief visual attention [10]. Other studies have used sensor-based technologies to monitor cyclists in real-time, capturing motion patterns. However, these studies are focused especially for maneuver prediction and abnormal driving [11], [12], while their potential for distraction analysis remains less explored.

Earables, wearable devices specifically designed to be worn in or around the ear, have integrated sensors such as inertial measurement units (IMUs). They offer precise head tracking capabilities, ergonomic design and surpass traditional wearables like smartphones and smartwatches due to fixed placement, minimizing motion artifacts. While eye tracking devices or smart glasses can also be used, they often suffer from ocular discrepancies like sunlight, sunglasses, etc., and head pose is easier to detect and said to have a direct correlation with eye gaze direction [13]. Earables are lightweight, unobtrusive and comfortable. They are also ubiquitous, commonly used when people are cycling, thereby enabling precise measurement of head movements [14], [15].

This paper explores the possibilities of the OpenEarable

platform [16], building on the potential of earable technology. The paper focuses on employing real-time head-movement analysis to identify and categorize cyclists' visual distractions. Visual distractions can be identified by monitoring a shift in gaze direction and head movements [17]. Prior studies in automotive contexts have shown that head movements can serve as a reliable indicator of visual distractions [18]. Leveraging this, the OpenEarable's integrated IMU sensors are used in a data-driven system to identify and classify head movement patterns related to visual distraction. Since the OpenEarable platform is open-source, further hardware and software customization is supported for scalable distraction monitoring solutions to be created for cycling scenarios.

The following research questions are addressed in this thesis in order to accomplish these goals:

- **RQ1:** Can visual distractions among cyclists be reliably captured from head movement data collected using IMU sensors in OpenEarable platform?
- **RQ2:** How to apply machine learning and deep learning models to IMU data obtained from earables for accurate and real-time detection of visual distractions in realistic cycling scenarios?
- **RQ3:** How can cyclist distraction detection models be efficiently deployed and evaluated within edge computing environments?

This paper presents a list of contributions aimed at answering the research questions. Firstly, a publicly accessible custom IMU-based dataset is created that captures the head movement patterns during distraction in dynamic, outdoor settings. Secondly, a new use case for earables is developed, highlighting their potential in distraction monitoring. This process involves data collection, preprocessing, feature extraction, and classification modelling utilizing both shallow and deep learning models. The models are designed under strict constraints ($<$ 5 MB) to enable efficient deployment on edge devices like Raspberry Pi. This 5 MB threshold is set as a practical upper bound during training to ensure the models remain both effective and lightweight, allowing further optimization for deployment. Among the models developed, CNN demonstrates moderate performance, with low model size and a substantial agreement to detect distractions, whereas in machine learning models, SVM has good accuracy measures with a fair Kappa score but with a fairly moderate model size.

While CNN and SVM models show good performance in distraction classification, deploying them efficiently on resource-constrained edge devices is critical for real-time monitoring. Microcontrollers in wearables and earables impose strict memory and computational limitations, creating a gap between model accuracy and practical deployability. Therefore, this work investigates the trade-off between model performance and deployability under edge constraints through a systematic evaluation. For SVM, which lacks standard quantization tools, feature set reduction techniques such as top-k feature selection and Principal Component Analysis (PCA) are examined to understand their impact on model size and classification performance. Use of an optimized SVM linear kernel called LinearSVC (a special SVM implementation from scikit-learn) is also evaluated to infer the performance tradeoff. For CNN models, optimization techniques like quantization, pruning and filter reduction are investigated to asses their effectiveness in reducing model size and inference time. A structured deployment framework is established to guide model selection and optimization decisions based on resource constraints and performance requirements. This deployment-centered evaluation provides real-world perspective on the accuracy-efficiency tradeoff and facilitates the development of earable-based distraction monitoring systems that are suitable for real-time, embedded deployment.

The remainder of the paper is structured as follows. Section II presents an overview of the previous studies on sensor-based techniques for cyclist monitoring and head gesture recognition using earables. Section III describes the experimental setup, data collection, preprocessing and model development for distraction detection. Section IV reports the results of the machine and deep learning model, while Section V discusses these results and outlines limitations and potential future work. Finally, the paper is concluded in Section VII.

II. Related work

This section reviews the recent advances in sensor-based cyclist behavior monitoring, especially focusing on maneuver predictions and abnormal driving. It then explores the studies that utilize earables as a solution for head movement tracking, and concludes by outlining the gap in existing research on distraction detection in cycling.

A. Sensor-based Monitoring

In the context of cycling, studies have used eye-tracking and camera-based systems for monitoring drivers' gaze, using features like fixation, glance duration, and saccade frequency, with a level of accuracy greater than 75% [19]. Eye-tracking glasses have also been used to predict cyclists' intent to change lanes [20]. CNNs trained on video data have classified cycling maneuvers, like passing, avoiding and sidewalk riding, with 82% accuracy [21]. One study labels visual distraction frame-by-frame using video, identifying instances when the head or gaze deviates from the road ahead [13]. Another study infers head pose and gaze direction to trigger alerts when attention is drawn away [22]. Since gaze detection is often hindered by sunglasses or lighting conditions, head pose becomes a more practical alternative [13], [20].

Inertial measurement units (IMUs) are accelerometer, gyroscope and magnetometer enabled and are highly effective in tracking head and body motion [12], [23]. Helmet-mounted IMUs have been used to classify cruising, turns, and right lane change at over 85% accuracy with a 4-second prediction interval [12]. IMUs mounted on bicycles achieved a turn prediction F1-score of 0.92 using a CNN-LSTM structure for capture of spatial and temporal motion patterns [11]. Intelligent helmets in motorcycling picked up head movements—upward, downward, leftward, and rightward—with 95.9%–99.1% accuracy

[24], while three-axis accelerometer-based systems coupled with FNNs and preprocessing methods such as SVM-SMOTE and t-SNE achieved 89.57% accuracy [25].

Smartphones, with built-in accelerometers, gyroscopes, and GPS, have been used extensively in sensing activities such as braking, turning, swerving, and lane weaving [26]–[28]. Random Forest and SVM machine learning models have achieved F1-scores of up to 0.90 [26]. GPS has been used to detect riding anomalies such as wrong-way cycling [29], and smartwatches have enabled motion tracking through hand gesture interpretation inside vehicles [30]. However, mounting sensors such as smartphones or IMUs on the bicycle results in significant noise from road vibrations, uneven surfaces, and shocks, which makes it difficult to accurately isolate motion patterns [26].

B. Earables

Earables provide a non-intrusive and comfortable framework for tracking motion patterns with good accuracy. Their ergonomic design encourages natural behavior without the constraints of helmets or bike-mounted sensors, making them universally acceptable across a wide variety of demographics [15]. Several studies have utilized the earables for motion-based behavior recognition [31], [32]. Devices like the eSense [15] with accelerometers and gyroscopes have also been used for detection and correction of forward head posture (FHP) [33]. Head gestures, including nodding and shaking, were successfully detected using the IMU sensors embedded in the eSense earable device [34], [35]. The studies mentioned have used accuracies and F1-score as their main metrics to show the reliability of the models developed for human activity recognition (HAR) activities. The classification of head movements achieved an F1-score of 88.24% [35]. Head movement classification (left, right and straight) was performed both in stationary and driving scenarios (car) using machine learning models like Random Forest and KNN yielding 96% accuracy [31]. Beyond head gestures, activities such as speaking, eating, walking, and posture changes have been recognized with over 90% accuracy using both machine learning and deep learning models like SVM, Random Forest and CNNs [36]. Classical machine learning models, combined with Dynamic Time Warping (DTW), were employed to classify boxing gestures, achieving an accuracy of 96% [32]. These findings highlight the potential of earables for real-time distraction monitoring in dynamic cycling environments.

C. Research Gap

One study explored cognitive distractions in an immersive virtual environment on the basis of physiological cues and head movements (e.g., Heart rate variability, electro-dermal activity) [6]. While it found correlations between cognitive load and reduced head movement, the experimental setup was expensive and artificial, and it did not explore the development of an end-to-end system to test real-world effectiveness.

To our knowledge, very few studies have specifically looked at the detection of distractions. Prior research for driving scenarios has demonstrated that head position measurements can reliably identify visual distractions when attention is deflected away from the road [17], [18], these results have not been sufficiently explored in the cycling domain. While existing research has shown that earables can effectively track head movements, few efforts have focused on building embedded framework that can recognize visual distractions based on motion patterns. Moreover, although some studies have explored activity recognition using earables, there is a clear gap in assessing the deployability of such models on constrained wearable devices. This study aims to fill these gaps by using the OpenEarable platform to explore how head movement data can be used as a proxy to detect visual distractions in cyclists and evaluate the feasibility of deploying visual distraction classification models on resource-constrained edge devices, thus aiming to reduce unsafe cycling behavior.

III. Methodology

This section describes the experimental setup, data collection using the OpenEarable device, preprocessing the collected data and the development of machine learning and deep learning models for distraction classification.

A. Experimental Setup

The experimental setup is designed to collect the head motion data from the participants using the OpenEarable device. The OpenEarable, powered by Arduino nano 33 BLE Sense, is equipped with 9-axis IMU sensor comprising an accelerometer, gyroscope and magnetometer and can stream data up to 50 Hz via Bluetooth low energy (BLE) [16]. Additionally, it includes ultrasound-capable microphone, an ear canal pressure sensor, a speaker and an RGB LED. Data logging can be performed using either a browser-based dashboard or a mobile app, which runs on a Nokia C32 smartphone to collect sensor data via Bluetooth Low Energy (BLE) during the experiment. An Akamduman Action camera (GoPro) is mounted on the bicycle's handlebar to continuously record the participant's face, which is later used as a ground truth for distraction labelling. The experiment is conducted in a secluded area with minimal to no traffic within the university campus to ensure participant safety while maintaining realistic riding conditions. The experimental route, as shown in Fig. 1, covers a distance of approximately 125.62 meters in length and about 27.50 meters in width.

B. Participants

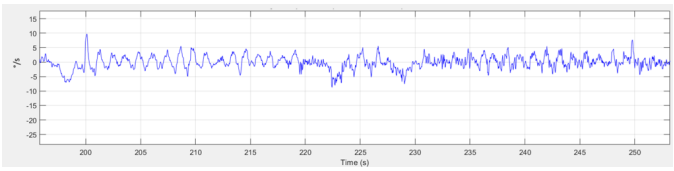
A total of 20 participants participated in this experiment (height in cm: 172.3 $\pm$6.7, weight in kg: 72.5$\pm$7.8, age in years: 24.9$\pm$ 1.9). All participants report no physical or neurological conditions that could impact their cycling performance and have a minimum of three years of riding experience. Each participant was informed about the purpose of the study and provided informed consent prior to participation. The experiment is conducted in accordance with the ethical guidelines approved by the Computer & Information Sciences (CIS) Ethics Committee of the University of Twente.

Fig. 1: Experimental route map

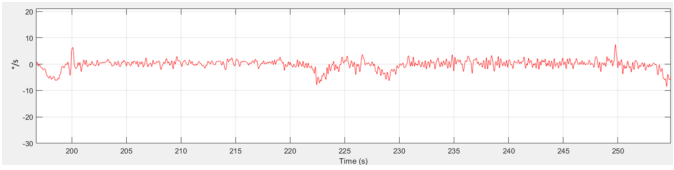
C. Data Collection

The data collection experiment to capture head movements is conducted under different conditions using the OpenEarable device. At the start, participants are asked to complete a consent form and questionnaire with demographic information and basic questions related to cycling. After placing the right OpenEarable on the participant and securing the mobile phone and the GoPro camera on the handlebar of the cycle, each participant is asked to complete five sessions. Each session consists of five laps along the predetermined route, covering a distance of 1545.9 meters. Before the start of each session, the OpenEarable is calibrated using the OpenEarable mobile App, which streams the 9-axis IMU data at 30 Hz using BLE. The participants are asked to perform head movements, such as up, down, left, and right, to see if the OpenEarable device is working as intended. Both the camera and the data recorder on the mobile phone are manually turned on by the researcher at the same time when the experiment begins.
The sessions are explained in detail below:

- **Session 1 and Session 2:** The participants cycle in the route at a leisure pace without any induced technological distractions. This is to monitor the visual distraction patterns under natural, normal driving behavior.
- **Session 3:** Participants listen to two low-tempo songs (Uptown funk by Bruno Mars and Sunflower by Post Malone) through a Bluetooth earbud placed in the left ear. The goal is to examine whether subtle auditory stimuli influence visual distraction behavior compared to the first two session.
- **Session 4:** Follow a similar structure as Session 3 but with two high-tempo songs (Starboy by The Weeknd and Bye Bye Bye by Nsync) to observe the visual distraction styles under more stimulating audio conditions.
- **Session 5:** A mock phone call is conducted between the participant and the researcher via Microsoft Teams. This scenario introduces a cognitively engaging distraction and is used to examine whether it affects the visual distraction profile compared to earlier sessions.



(a) Raw Gyroscope Z-axis Signal



(b) Filtered Gyroscope Z-axis Signal

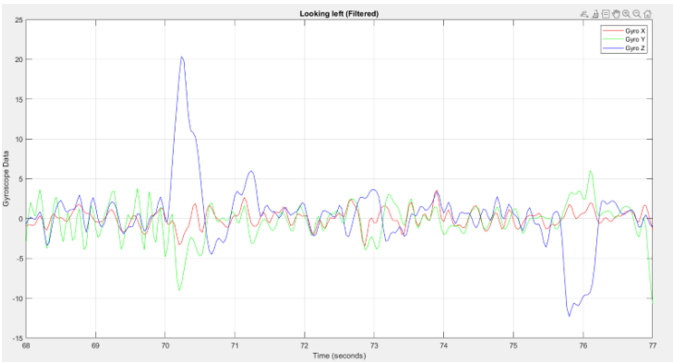
Fig. 2: (a) Raw signal showing wobbling motion around 200–220 seconds, and (b) filtered signal after removing the wobbling component

The list of questions used by the researcher during the phone call session is provided in Appendix A. This structured approach ensures that participants experience different distraction conditions, providing valuable data that is used for distraction analysis. At the end of each session, the data logged on the mobile phone is saved and uploaded to the cloud.

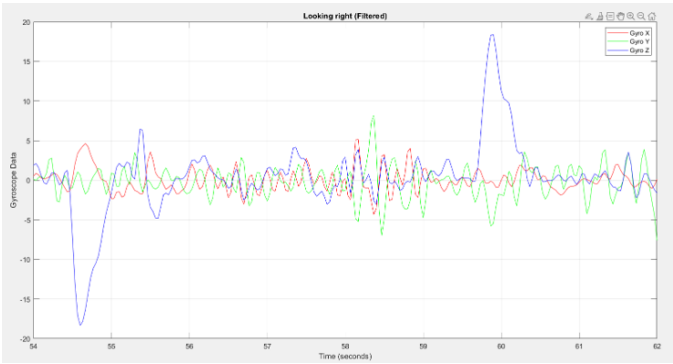
D. Data Pre-processing

The data collected from the OpenEarable device includes 9-axis IMU signals, sampled at 30 Hz (the highest sampling rate provided by the OpenEarable app). Magnetometer data are excluded from further analysis to avoid distortion caused by metallic interference and from other electrical devices, such as Bluetooth earbuds worn on the left ear [37].

To improve the quality of the signals and reduce artifacts, a two-step filtering process is applied to the raw inertial data collected from the OpenEarable device. Most participants exhibit wobbling head motion, especially during straight-line cycling. The wobbling head motion is purely personal and contributes significantly to overall head movements. To remove these wobbling motions, the frequency of the wobbling motion (seen more clearly on gyroscope of Z-axis) is determined separately for each participant and each session. A notch filter is then designed and applied around the detected wobbling frequency, effectively eliminating the wobbling component. Following this, a second-stage filtering is performed using a low-pass filter with a cutoff frequency of 6 Hz on all axes of the accelerometer and gyroscope. This helps to remove remaining high-frequency noise while retaining meaningful motion patterns. This is illustrated in Fig. 2, which compares raw and filtered data.



(a) Gyroscope signal for looking left



(b) Gyroscope signal for looking right

Fig. 3: Gyroscope signals capturing head gestures: (a) turning left and returning to center, and (b) turning right and returning to center.

Labelling of the data is manually performed using the video captured from the camera. Each session video is reviewed frame by frame and visual distraction events are annotated and time-aligned with the IMU data. Head movements, such as looking left and right, can be clearly observed in the gyroscope data. A distinct positive peak is observed in the Gyro-Z axis when participant turn their head to the left, followed by a negative peak as the participant returns to the forward-facing position. For looking right, the pattern is reversed, as shown in Fig. 3. Similarly, patterns for looking up and looking down can be observed in the Gyro-Y axis.

To classify between distraction and non-distraction events, the duration for which the participants' head is not facing forward is used as a primary criterion. The following labeling system is applied using a three-second threshold, based on the guidelines provided by research on real-world driver behavior [38].

- *Distracted Looking Left (DLL)* and *Distracted Looking Right (DLR)*: When the participants head is not facing the front road for more than 3 seconds.
- *Non-distracted Looking Left (NLL)* and *Non-distracted Looking Right (NLR)*: Head turns that last less than 3 seconds.
- *Looking Down (LD)*: Considered as a distracted event signaling visual fixation on phone screen.

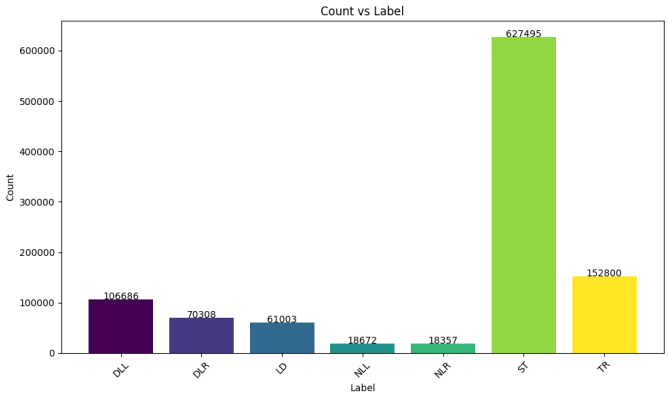


Fig. 4: Label distribution

- *Straights (ST)*: Period when participants focus on the road without any head movements.
- *Turn Right (TR)*: Captures intentional right turns during cycling, as the route only includes right turns.

An event is labelled as a visual distraction not just based on head orientation, but also based on the participant's gaze. For instance, if a participant turns their head but their eyes remain focused forward, such events are not labelled as visual distractions. It is only when both the head and eye direction deviate for more than 3 seconds, the event is labelled as a visual distraction. This ensures that the label identifies meaningful attention shifts and avoids classifying intentional, momentary head movements. A summary of label distribution across the dataset is presented in Fig. 4. The proportion of total distraction events across all five sessions from all participants is shown in Fig. 5.

The preprocessed data is then segmented using a classical windowing approach. Three window sizes of 2 seconds, 5 seconds and 7 seconds are considered with an overlap of 50% for each window size. The window sizes are chosen because

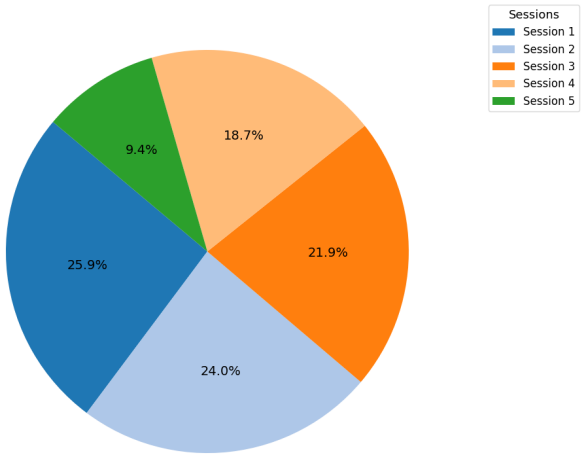


Fig. 5: Distribution of distraction events across sessions

TABLE I: IMU Feature Descriptions

Feature	Description
Mean	The average value of the IMU readings (x,y and z axis) over a time window.
Standard Deviation	Measures variability or dispersion of IMU values around the mean within time window.
Range	Difference between the maximum and minimum values within the window for each IMU axis.
Root Mean Square (RMS)	Reflects signal magnitude, computed as square root of the mean of squared values.
Maximum	Largest value recorded in each IMU axis and magnitude during the window.
Minimum	Smallest value recorded in each IMU axis and magnitude during the window.
Energy	Sum of squares of IMU values in a window, indicating the overall intensity of movement.
Mean Absolute Deviation (MAD)	Mean of absolute differences between values and their mean, capturing spread.
Band Power	Power of the signal in a frequency band (e.g., 0.5–3 Hz), estimated using Welch's method.
Spectral Entropy	Measures the complexity of the signal by analyzing the power distribution in the frequency spectrum.
Interquartile Range (IQR)	Range between 75th and 25th percentiles, robust against outliers.

different participants exhibit different distraction patterns, with some being distracted for more than 7 seconds. A majority labelling method is applied to assign a single label for each window. If a window contains multiple labels (e.g. both distracted and non-distracted events), the label that appears most frequently within that window is chosen as the representative label for that window.

To train classical machine learning models, various features are extracted from windowed data. The selected statistical features are shown to support accurate classification and prediction of IMU data [24], [28]. The features used are shown in Table I.

E. Model Implementation

Various machine learning and deep learning models are trained to classify visual distractions. To ensure a fair and consistent comparison across models, all machine learning and deep learning models are developed with common constraints: keeping the final model size under 5 MB while maintaining an acceptable level of accuracy. The 5 MB threshold is not the final deployment size, but a practical upper bound used during model development to design a compact and performance-efficient architecture. Therefore, the aim is to identify high-performing models that are already reasonably lightweight and can be further optimized for deployment. To achieve this, a GridSearch is conducted for all models to select the hyperparameters for optimal performance. The GridSearch is a hyperparameter tuning technique that tests all possible combinations within a predefined grid to find the best-performing parameters. However, the hyperparameters selected using GridSearch resulted in a model size exceeding 5 MB. Therefore, the hyperparameters are further manually tuned to decrease the model size to less than 5 MB, while also not compromising on performance, thereby finding a balance between the two.

The extracted features are used to train various classical machine learning models. The models include *Support Vector Machine (SVM), XgBoost, Random Forest (RF)* and *K-Nearest Neighbors (KNN)*, all of which are implemented using the scikit-learn library [39]. The models are selected due to their effectiveness in classifying IMU-based data. For SVM, a Linear kernel is used, which balances the model complexity and generalization while keeping the model lightweight. Since the dataset is imbalanced, computed weights are used in XgBoost model during training. This adjusts the loss function by assigning higher weights to the samples from underrepresented classes. Random Forest is another ensemble method that aggregates the predictions of multiple decision trees, which reduces overfitting, and KNN is a simple classification model that assigns the class to a sample based on the majority class of its nearest neighbors. The parameters are listed in Table II.

Deep learning pipelines like *Convolutional Neural Networks (CNNs), CNN-LSTM (CNN combined with Long Short-Term Memory), DeepConv LSTM* [40] and *Temporal Convolutional Networks (TCNs)* are considered and implemented using the Tensorflow Keras API. The input shape is window_size $\times$ num_features, where the num_features is 6 and window_size is 60, 150 and 210 for 2 seconds, 5 seconds, 7 seconds windowing respectively. The CNN model consists of two 1D convolutional layers followed by a max-pooling and a dense output layers to capture spatial features. The combination of a CNN and LSTM for CNN-LSTM allows the model to extract both spatial and temporal features. The DeepConvLSTMs are widely used in HAR (human activity recognition) and extends their model by stacking additional convolutional layers before the LSTM layers, allowing the model to learn long-range dependencies for more complex feature extraction

TABLE II: Parameters for Machine Learning Algorithms

ML Models	Parameters
Support Vector Machine (SVM)	Regularization parameter (C): 1 Kernel type: linear Kernel coefficient (gamma): 0.001
XgBoost	Number of trees (n_estimators): 200 Maximum tree depth (max_depth): 6 Learning rate (learning_rate): 0.05 Subsample ratio (subsample): 0.8
Random Forest (RF)	Number of trees (n_estimators): 75 Maximum tree depth (max_depth): 10 Minimum samples to split (min_samples_split): 5 Minimum samples per leaf (min_samples_leaf): 5
k-Nearest Neighbors (KNN)	Neighbors (n_neighbors): 11 Weight function: distance Distance metric: euclidean

TABLE III: Deep Learning Model Architectures and Hyper-parameters

Model	Architecture Details
CNN	- Two 1D Conv layers (64 filters, kernel size 3, ReLU activation) - MaxPooling1D (pool size 2) - Dropout (rate 0.5) - Dense (100 units, ReLU) + Output Layer
CNN-LSTM	- 1D Conv layer (64 filters, kernel size 3, ReLU activation) - MaxPooling1D (pool size 2) - LSTM (32 units, dropout 0.2, L2 regularization) - Output Layer
DeepConv-LSTM	- Four 1D Conv layers (64 filters, kernel size 5, same padding, L2 regularization) - LSTM (128 units) $\rightarrow$ LSTM (64 units), both with dropout 0.3 and recurrent dropout 0.3 - Dense (64 units, ReLU) + Dropout (0.5) + Output Layer
TCN	- TCN block: 64 filters, kernel size 5, dilations [1, 2, 4, 8], dropout rate 0.3 - Output Layer

[40]. TCNs are used as an alternative to LSTMs, using dilated convolutions to capture long-range temporal dependencies efficiently. TCNs offer a faster and more efficient approach to modeling temporal patterns without the use of recurrent layers. The hyperparameters selected for each deep learning model are shown in Table III. All models are optimized using Adam optimizers and sparse categorical cross-entropy for multi-class classification, with a batch size of 64. Although training is set for 100 epochs, early stopping with a patience of 5 epochs helps the models converge faster and avoid overfitting.

F. Evaluation

A 75:25 train-test split is used to train and evaluate both the machine learning and deep learning models. This split provides a balanced distribution of data for training and ensures sufficient samples for evaluating generalization performance. In addition to Train-Test split, Group K-Fold and Leave-One-Subject-Out (LOSO) cross-validation techniques are employed to assess the robustness of the models across different participant groups. Train-test split provides a baseline testing on new participants unseen during training, Group K-Fold (with K = 5) offers more stable average performance by dividing the participants into 5 groups and iteratively using each group as test set. Leave-One-Subject-Out (LOSO) is used to rigorously test generalization by testing on one entirely unseen participant at a time.

The models are evaluated using several key performance metrics. The *F1-score* provides a balance between precision and recall, especially beneficial for imbalanced datasets. Since our dataset is highly imbalanced, traditional accuracy may present a biased view of model performance, therefore *Cohen's Kappa score* is evaluated as a chance-corrected measure that presents the level of agreement between the predicted and true labels. The formula for Cohen's Kappa is is shown in Equation 1, where the observed agreement (p_o) is the proportion of instances that are classified the same by both the model and the ground truth, and the expected agreement (p_e) is the proportion of instances that would be classified the same purely by chance. The interpretation of Kappa scores by Landis & Koch [41] is given in Table IV. *Model size* is considered to check storage and deployment efficiency, and *inference time* measures how quickly the model can make predictions, which is crucial for real-time applications. Together, these metrics offer a comprehensive understanding of the models' performance and effectiveness.

TABLE IV: Interpretation of Cohen's Kappa Score

Kappa Value	Interpretation
> 0.8	Almost Perfect Agreement
> 0.6	Substantial
> 0.4	Moderate
> 0.2	Fair
$0 - 0.2$	Slight
< 0	Poor (no agreement)

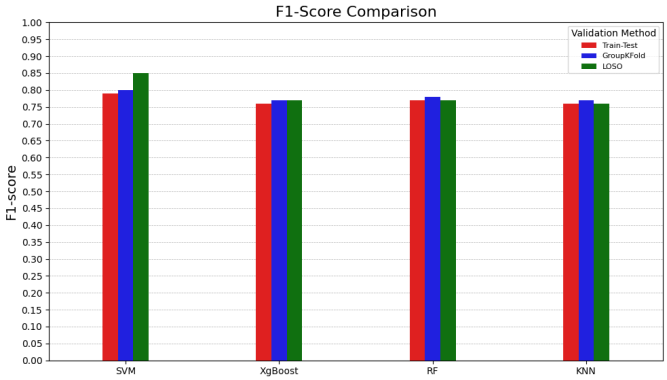


Fig. 6: F1-score of Machine Learning models

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{1}$$

IV. Results

A. Machine Learning Models

Several machine learning models have been trained and compared to determine the most suitable one for the dataset. Among the window sizes tested, the 7-second window with 50% overlap delivers the best balance of performance, accuracy, model size, and training time. On the other hand, the 2-second and 5-second windows require more data windows, resulting in larger model sizes and longer training times. For simplicity, the results presented here focus on the 7-second window, as discussing the other two window sizes would take up unnecessary space without providing significant additional value.

Fig. 6 presents the comparison of the F1-score across the ML models. SVM outperforms under LOSO (0.85) compared with Train-Test (0.79) and Group K-Fold (0.80), indicating

TABLE V: Inference Time (ms) for Machine Learning Models

ML Models	Inference Time (ms)		
	Train-Test	GroupKFold	LOSO
Support Vector Machine (SVM)	0.3058	0.4205	0.3709
XgBoost	0.0025	0.0041	0.0058
Random Forest (RF)	0.0121	0.0109	0.0127
k-Nearest Neighbors (KNN)	0.0182	0.0186	0.0189

TABLE VI: Cohen's Kappa Scores for Machine Learning Models

ML Models	Cohen's Kappa Score		
	Train-Test	GroupKFold	LOSO
Support Vector Machine (SVM)	0.43	0.42	0.59
XgBoost	0.44	0.46	0.44
Random Forest (RF)	0.37	0.38	0.37
k-Nearest Neighbors (KNN)	0.35	0.36	0.35

very good generalization to unseen participants. This is because SVM's margin-based decision rule and regularization prevent overfitting of participant-specific patterns. However, the F1-scores of Random Forest and XGBoost are relatively stable but poor (around 0.76–0.78), possibly because they are limited by their shallow depth and the small number of trees, owing to the constraint on model size. These ensemble models are good at detecting training patterns but can likely perform poorly in generalizing across diverse participants. KNN performs worst (0.76–0.77), as it lacks a learning process and relies on distance-based classification, which is susceptible to variations among individuals and disadvantaged in high-dimensional feature spaces.

The inference time of each model is shown in Table V. based on the time taken to classify a single 7-second IMU window, providing a realistic measure of latency for real-time deployment. XgBoost exhibits the fastest inference times, around 0.005 ms, due to its efficient gradient boosting structure. Whereas, SVM has moderate inference time (< 0.5 millisecond) as prediction requires computing multiple support vectors. Random Forest (RF) and KNN have similar inference times.

The F1-score against model size trade-off plot in Fig. 7 illustrates the trade-off between performance and memory consumption for each of the machine learning models. The SVM model has the largest model size of 3.65 MB. This increase in size can be explained since SVM stores numerous support vectors needed for generalization. But it is still the top-performing model compared to all other machine learning models. Whereas, XgBoost and Random Forest models have similar model sizes (around 3.3 MB) reflected by multiple trees in ensemble methods but has low F1-scores when compared to SVM. KNN has the smallest model size of 1.96 MB and lowest F1-score (0.76) since it does not learn explicit parameters but rather stores the feature vectors.

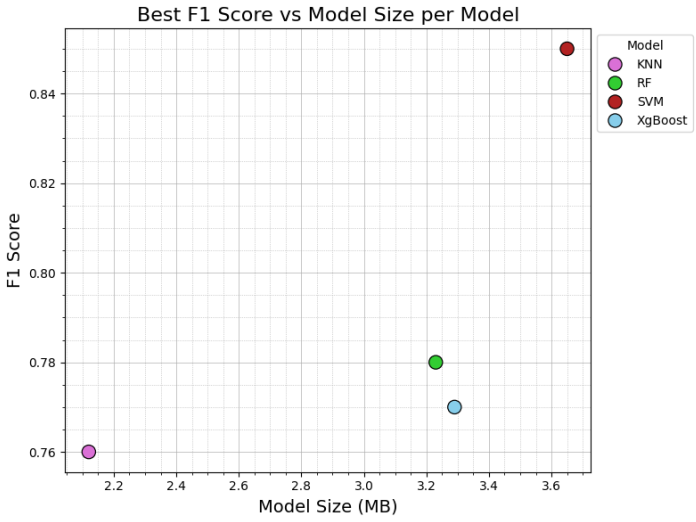


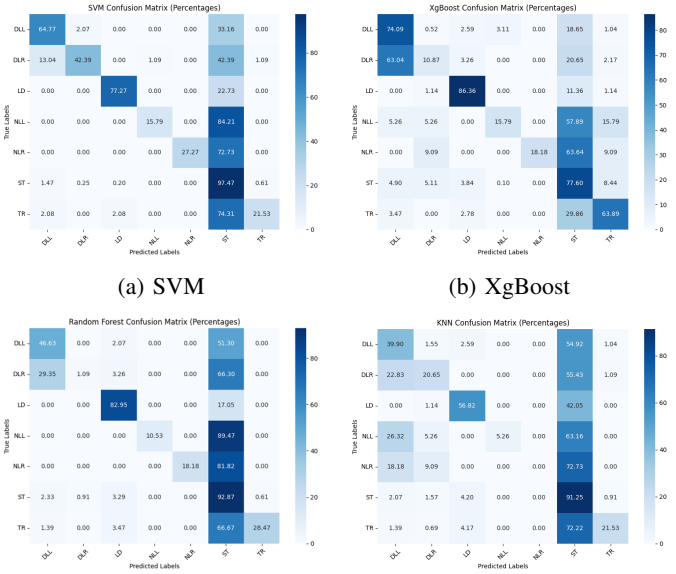
Fig. 7: F1-Score vs Model Size for ML models

In addition to the standard performance metrics, Cohen's Kappa score is presented in Table VI. The SVM model under LOSO validation has a kappa score of 0.59, indicating moderate classification reliability. This suggests that the model is reasonably reliable at classifying distraction events, even in the presence of class imbalance. However, Random Forest and KNN models show slight agreement with kappa scores less than 0.40, meaning their predictions are only slightly better than random guessing. To further examine the classification performance, confusion matrices of each machine learning model are provided in Fig. 8. These provide a detailed view of how each model distinguishes between different distraction classes. Many misclassifications are observed for Straights (ST), which is due to the class imbalance in the dataset, where ST significantly outnumbers the other classes, leading the model to favor it more during prediction. Additionally, NLR and NLL are often confused with ST, likely because these involve minimal head movements and largely resembles straight-head motion. These trends indicate the need for more balanced data sets or augmentation in order to improve class-specific sensitivity.

B. Deep Learning Models

Deep learning models, such as CNN, CNN-LSTM, DeepConv-LSTM, and Temporal Convolutional Networks (TCNs), are evaluated. All results in this section are presented using LOSO validation and a 2-second window with 50% overlap, as this configuration provides the best performance across all deep learning models in terms of efficiency and real-time applicability. For simplicity, the results presented here focus only on 2-second windowing.

Fig. 9 shows the F1-scores of the deep learning models. CNN provides the highest accuracy of 0.87, indication reliable performance in accurately identifying distraction classes despite class imbalance. This is explained by its architecture, which is adequately capable of extracting spatial features from the IMU data using convolutional layers. TCN follows with an F1-score of 0.80. Its highlight is the use of dilated causal



(a) SVM (b) XgBoost

(c) Random Forest (d) KNN

Fig. 8: Confusion matrices of Machine Learning models

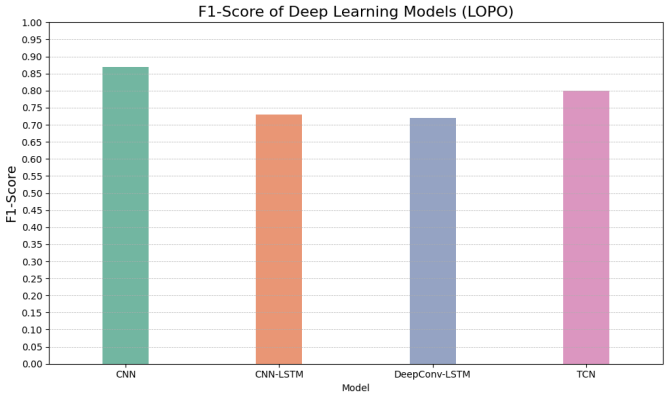


Fig. 9: F1-score of Deep Learning models

TABLE VII: Inference Time for Deep Learning Models

DL Models	Inference Time (ms)
Convolutional Neural Networks (CNN)	0.0703
CNN-LSTM	0.1590
DeepConv-LSTM	0.7276
Temporal Convolutional Network (TCN)	0.1870

convolutions, which effectively capture long-range temporal dependencies without relying on recurrent layers. However, CNN-LSTM and DeepConv-LSTM models have lower F1-scores of 0.73 and 0.72, respectively. These models introduce recurrent layers (LSTMs), which add complexity and increase the number of trainable parameters within the model. With minimal training data per participant and significant inter-participant variability, these models are more prone to overfitting.

The per-window inference time (2-second window) for deep learning models are shown in Table VII. CNN has the fastest inference time 0.0703 ms, which uses only convolutional and dense layers with relatively few parameters. CNN-LSTM has an inference time of 0.1590 ms, which is slightly slower due to the inclusion of LSTM layers. Whereas, the DeepConv-LSTM demonstrates the longest inference time of 0.7276 ms as would be expected due to its complicated structure of having deeper stack of convolutional layers followed by LSTM layers. All the deep learning models has achieved sub-millisecond inference latency and are suitable for real-world applicability to classify distractions swiftly.

To show the trade-off between model size and prediction performance, Fig. 10 shows F1-Score vs. Model Size (MB) for all models. CNN achieves the best balance with the largest F1-Score of 0.87 with a moderate model size (0.74 MB). TCN is also comparable with an F1-Score of 0.80 and small size (0.56 MB). Whereas, DeepConv-LSTM generates the largest model (0.83 MB) and poorest F1-Score of 0.73, revealing increased depth does not necessarily translate to good performance. CNN-LSTM generates the smallest model (0.03 MB) but also poorest F1-Score. This poor performance is due to the fact that the model had been highly compressed during training in order to keep the model size under the provided constraint and hence cause its inability to capture meaningful temporal patterns. It is also observed that deep learning models have smaller model sizes compared to conventional machine learning models. This is because deep learning models learn low-dimensional features internally from raw IMU data, as opposed to machine learning models which receive high-dimensional statistical features as inputs. The traditional models such as SVM save numerous support vectors and ensemble algorithms such as XgBoost and Random Forest saves numerous decision trees, which once again makes them bigger in size. Deep learning models save only learned weights, thus significantly smaller model sizes.

Cohen's Kappa score is reported for all the deep learning

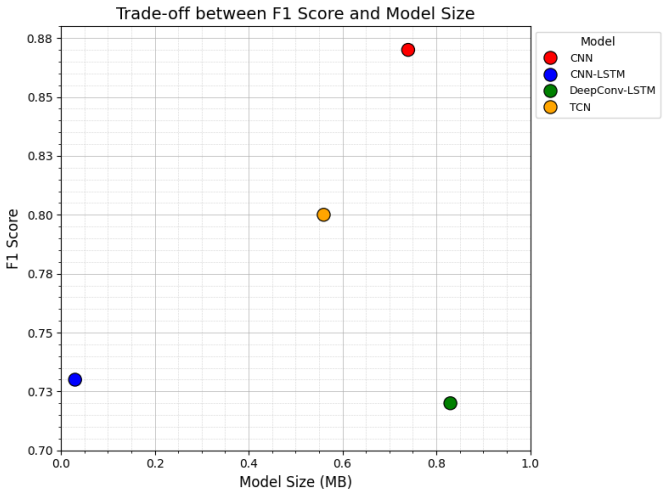


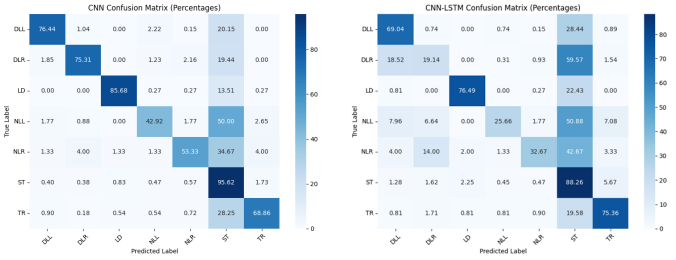
Fig. 10: F1-Score vs Model size for DL models

TABLE VIII: Cohen's Kappa Score for Deep Learning Models

DL Models	Kappa Score
Convolutional Neural Networks (CNN)	0.74
CNN-LSTM	0.54
DeepConv-LSTM	0.49
Temporal Convolutional Network (TCN)	0.57

models in Table VIII. The Kappa score of the CNN model is 0.74, which falls under substantial agreement, and therefore indicates high correspondence between predictions made by the model and the actual labels beyond chance. CNN-LSTM and TCN models have Kappa scores of 0.54 and 0.57, respectively, both being under moderate agreement, and therefore the predictions are quite reliable but not strongly consistent. The DeepConv-LSTM model attains the Kappa measure of 0.49, which indicates fair agreement, suggesting only a low level of prediction reliability. To further evaluate the classification performance, the confusion matrices for each deep learning model are presented in Fig. 11. Similar to machine learning results, a large number of misclassifications are observed for Straights (ST) because of the class imbalance in the data where straight riding instance dominates the label distribution. CNN demonstrates better separation between all distraction classes (DLL, DLR and LD), whereas other models like CNN-LSTM and DeepConv-LSTM misclassified the distraction classes, indicating their difficulty in distinguishing distractions and naturally occurring head turns.

Overall, the CNN model has a best performance of all the models trained with F1-score of 87% and a substantial Kappa score of 0.74. In addition, CNN has compact model size of 0.74 MB, making it highly suitable for edge deployment.



(a) CNN (b) CNN-LSTM

(c) DeepConv-LSTM (d) TCN

Fig. 11: Confusion matrices of Deep learning models

Among classical machine learning models, SVM demonstrates reasonable performance with a fair Kappa score (0.59). However, the model size of 3.65 MB is large when compared to the CNN model, as previously discussed , due to the storage of support vectors and reliance of statistical features.

C. Deployment

To assess the real-world applicability of the distraction classification models, the best-performing machine learning and deep learning models are prepared for deployment to evaluate the feasibility on earable devices with resource-constrained microcontrollers. The OpenEarable is powered by an Arduino Nano 33 BLE Sense microcontroller has 1MB of flash memory and 256 KB of SRAM for runtime operations. Therefore, the deployment objective is to reduce the model size to less than 256 KB, while reducing inference speed and maintaining actual performance accuracy. However, for this implementation, the actual deployment is on a Raspberry Pi 4, which has sufficient computing and memory capability to extensively test model performance, evaluate optimization techniques, and simulate edge deployment scenarios. This platform serves as an intermediate environment prior to deployment to microcontrollers like the Arduino Nano 33 BLE Sense, enabling controlled benchmarking of inference latency, memory usage, and model behavior under constrained conditions.

For convenience of deployment on the Raspberry Pi 4, the trained models and the optimized variants are exported in suitable formats. Machine learning models such as SVM are serialized using the Pickle (.pkl) format, while deep learning models are saved as TensorFlow Lite (.tflite) format through the TensorFlow Lite converter. These formats allow for efficient loading as well as execution on the edge devices. Following deployment, inference takes place with the last five participants' test data, in line with the evaluation procedure followed in previous sections. This follows that there is a consistent comparison between the deployed model's output and the baseline results gathered during the first round of experimentation.

1) SVM Deployment: The SVM model is chosen as it demonstrated the best performance among all the other machine learning models. However, the baseline SVM model trained using all extracted features (96 features), has a model size of 3.65 MB, which exceeds the 256 KB threshold. SVM models do not support quantization in scikit-learn, therefore other methods like feature selection, dimensionality reduction and using linearSVC kernel are explored.

- **Feature selection using SelectKBest**: SelectKBest uses univariate statistical test (ANOVA F-value), which ranks features based on the correlation with the target variables. Irrelevant or redundant features are removed, thereby reducing the model size and the computation complexity.
- **Principal Component Analysis (PCA)**: A dimensionality reduction technique which transforms the original feature space (96 features) into a lower-dimensional space

(new 16 features) while retaining high variance as possible. This helps reduce model size and inference time.

- **LinearSVC**: LinearSVC is a linear classifier which uses the libnear library instead of default libsvm. It is specifically tailored for high-dimensional data and offers a memory efficient implementation since it does not store support vectors (contrary to default linear SVM) and thereby greatly reducing the model size and inference time.

The deployment results in Table IX shows the trade-offs between model size, performance and resource usage. The baseline SVM model with the best performance has a model size of 3.65 MB exceeding the model constraints.

Feature selection technique SelectKBest does reduce the model size to half (1.25 MB), but comes at the cost of reduced performance around 5%. Principle component analysis offers similar accuracy and lower inference time (0.97 ms).

LinearSVC significantly reduces the model size to 173 KB and achieves the fastest inference time of 0.13 ms, while remaining within the 256 KB deployment constraint. Although the CPU usage of LinearSVC is more than 90%, it is important to note that this measurement is based on single-core execution since Rpi (Quad core ARM Cortex-A72) has 4 cores.

To conclude LinearSVC stands out as the most suitable option since it provides the best balance between deployment feasibility and classification performance.

2) CNN Deployment: Among the deep learning models, CNN achieved the highest performance with a strong F1-score. Also, the model size is relatively small for a deep model, mainly due to the parameters chosen while training. However, the baseline model (0.74 MB) still exceeds the 256 KB memory constraint required for deployment. Various deployment optimization and compression techniques, such as quantization and pruning, can be applied to deep learning models to reduce memory footprint and inference latency [42]. The techniques explored in this implementation are as follows:

- **Post-Training Quantization (PTQ)**: Quantization reduced the precision of weights to 8-bit (INT8) or 16-bit floats (FLOAT16) from 32-bits. This can be done after a model is fully trained, therefore no need to retrain the model during quantization.
- **Quantization Aware Training (QAT)**: QAT simulates low precision (INT8) operations during the training phase by inserting fake quantization nodes. This allows the model to learn and adapt to quantization effects, making it more robust compared to models quantized post-training (PTQ). QAT optimizes the model's weights and activations under quantized conditions, making it especially effective for preserving performance in small or sensitive models intended for edge deployment.
- **Filter reduction**: The number of convolutional filters in the trained CNN model is reduced from 64→32→16→8 to lower the model size and parameter count. Although this simplifies the model architecture, it may degrade the performance if performed aggressively.
- **Pruning**:Unstructured pruning is an effective technique for reducing the computation and memory requirements of deep learning models. Elimination of less important weights (weights with lower magnitudes) adapts the model to be more efficient and lightweight without much loss of performance.

The inference results in Table X shows a comprehensive comparison off CNN variants optimized for deployments. The baseline CNN model has strong performance metrics but exceeds 256 KB limit. Post-training quantization (PTQ) using FLOAT16 and INT8 reduces the model size to 0.37 MB and 0.19 MB respectively with minimum performance degradation. However, the FLOAT16 variant does not reduce the memory size less than 256 KB and the there is no inference time speed up since FLOAT16 arithmetic is not natively supported by Raspberry Pi. In contrast, INT8 quantization has model size less than 256 KB and inference time improved from 1.0784ms to 0.7798 ms, achieving a $\sim 27.7\%$ speedup.

It is noteworthy that quantization-aware training (QAT) enhances the INT8 model by simulating quantization operations during the training process, resulting in an improved F1-score of 0.89. The inference time and RAM usage are also reduced, while the model size is the same as the PTQ INT8 quantized variant.

The filter reduction technique can be applied when the hardware lacks support for low-precision data type operations, such as FLOAT16 and INT8. By reducing the number of convolutional filters, the number of trainable parameters and computations is significantly lowered, thereby greatly reducing both the model size and latency. However, this leads to a performance drop of approximately 5-6%, and retraining the whole model is required to adapt the model to the reduced capacity. For the CNN model developed, a filter size of 16 provides the best balance between performance, model size and inference time.

Unstructured pruning is performed using TensorFlow's pruning API, with a goal of pruning 50% of the weights in the convolutional and dense layers. Polynomial Decay Pruning (PDP) schedule gradually increase the pruning from 0% to 50% over the course of training, rather than pruning weights abruptly. Although it reduces the number of active parameters, it does not reshape or compress tensors. Therefore, the model size stays the same (0.74 MB). Also, because the Raspberry Pi does not support sparse computations, there is no noticeable improvement in inference time compared to the baseline. However, when pruning is combined with INT8 quantization, it does lead to a faster inference time of 0.6170 ms and a reduced model size of 0.19 MB, all while maintaining the same level of accuracy.

The deployment methodology is guided by a systematic process presented in Fig. 12. The flowchart outlines the systematic approach for selecting and optimizing suitable models for deployment on resource-constrained earable devices. In the absence of a pre-trained high capacity DL models, lightweight models like SVM explores way to reduce the model size by reducing feature space either using feature

TABLE IX: Comparison of SVM Deployment Variants on Raspberry Pi 4

Configuration	Model Size (MB)	Inference Time (ms)	RAM Usage (KB)	CPU Usage (%)	F1-score
Baseline	3.65	3.2613	192.71	83.76	0.85
SelectKBest (Top-16)	1.25	1.2279	210.59	90.39	0.77
PCA (16 features)	1.28	0.9678	189.32	83.62	0.77
LinearSVC	0.173	0.1330	186.97	90.47	0.83

TABLE X: Comparison of CNN Deployment Variants on Raspberry Pi 4

Configuration	Model Size (MB)	Inference Time (ms)	RAM Usage (MB)	CPU Usage (%)	F1-score
Baseline	0.74	1.0784	424	64.53	0.87
Float16 Quantization	0.37	1.1956	412	70.44	0.87
INT8 Quantization	0.19	0.7798	455	63.44	0.86
QAT + INT8	0.19	0.7034	302	60.54	0.89
Filter Size 32	0.36	0.4712	412	73.44	0.81
Filter Size 16	0.18	0.1631	369	65.33	0.83
Filter Size 8	0.09	0.0990	421	73.44	0.81
Pruning (50%)	0.74	1.0775	382	67.53	0.87
Pruning (50%) + INT8	0.19	0.6170	449	55.00	0.86

selection (SelectKBest) or dimensionality reduction (PCA). Alternatively, in the case of SVM, LinearSVC can be used as a space-efficient alternative, more robust than the feature dimensionality reduction techniques. For deep networks like CNN, techniques like quantization, pruning and filter reduction can be used, depending on what the target hardware supports. This systematic approach introduces a definite process to select the right model combination and optimization techniques, making the deployment decisions both effective and aligned with the constraints of the embedded platform.

In conclusion, both post-training INT8 quantization and quantization-aware training (QAT) with INT8 stand out as the most practical deployment options for this work. They strike a good balance between accuracy, compact model size, and fast inference. With their size reduced to just 0.19 MB and inference times well under 1 millisecond, these models meet the strict memory and speed requirements typical of microcontroller-based systems. While the deployment tests were carried out on a Raspberry Pi, the final goal is to run these models on the OpenEarable powered by Arduino Nano 33 BLE Sense. Since the OpenEarable also supports low-precision INT8 operations, the models are well-suited for real-time inference in embedded earable applications.

V. Discussion

A. Data analysis

In the data collection experiment, visual distraction activity varies notably across different sessions. For most participants, session 5 (phone call session) results in fewer visual distraction instances compared to the other sessions as seen in Fig. 5. A likely reason is that during a phone call, participants are cognitively engaged in conversation, leaving them with less mental capacity to visually wander. The distractions that do occur in this session often happen when participants pause to think about how to respond, which is relatively infrequent.

Session 1 and session 2, which represent normal riding conditions without any imposed distractions, show similar levels of visual distraction. This is expected, as participants tend to direct their gaze to specific areas of interest along the route and will repeat these visual patterns for return laps.

Sessions 3 and 4, listening to low-tempo and high-tempo songs, respectively, demonstrate varying levels of visual distraction in participants. Most participants show nodding behavior while listening to music, particularly in session 4, as a natural response to enjoying the rhythm of the songs. These nods introduce slight motion artifacts in the earable data which is later removed in the preprocessing stage.

These observations answer **RQ1**, indicating that visual distractions between cyclists can be consistently measured via head movement data obtained from the OpenEarable platform. The dataset obtained from 20 participants illustrates that IMU signals yield distinguishable patterns throughout distraction periods, validating their use for detecting attention states.

B. ML and DL results

SVM achieved the highest F1-score (85%) among all the machine learning models discussed and have a near substantial Kappa score (0.59) thus showing its ability to define boundaries for accurate distraction classification. XgBoost provides comparable performance with faster inference time but has low Kappa scores. Random Forest and KNN has poor Kappa scores, indicating they are not suitable to classify distractions correctly in an imbalanced dataset. This is due to the way the Random Forest and KNN models are fine tuned to have low model size less than 5 MB.

In deep learning models, CNN achieved the highest F1-score (87%) and a substantial Kappa score (0.74) with a moderate model size of 0.74 MB. This shows CNN are excellent for handling temporal and spatial features from the 2-second windowed IMU data. However, deeper architectures like CNN-LSTM and DeepConv-LSTM struggles to perform well when compared to simpler CNNs due to the potential of overfitting due to smaller dataset thus reducing generalization. Especially, DeepConv-LSTM exhibits longest inference time

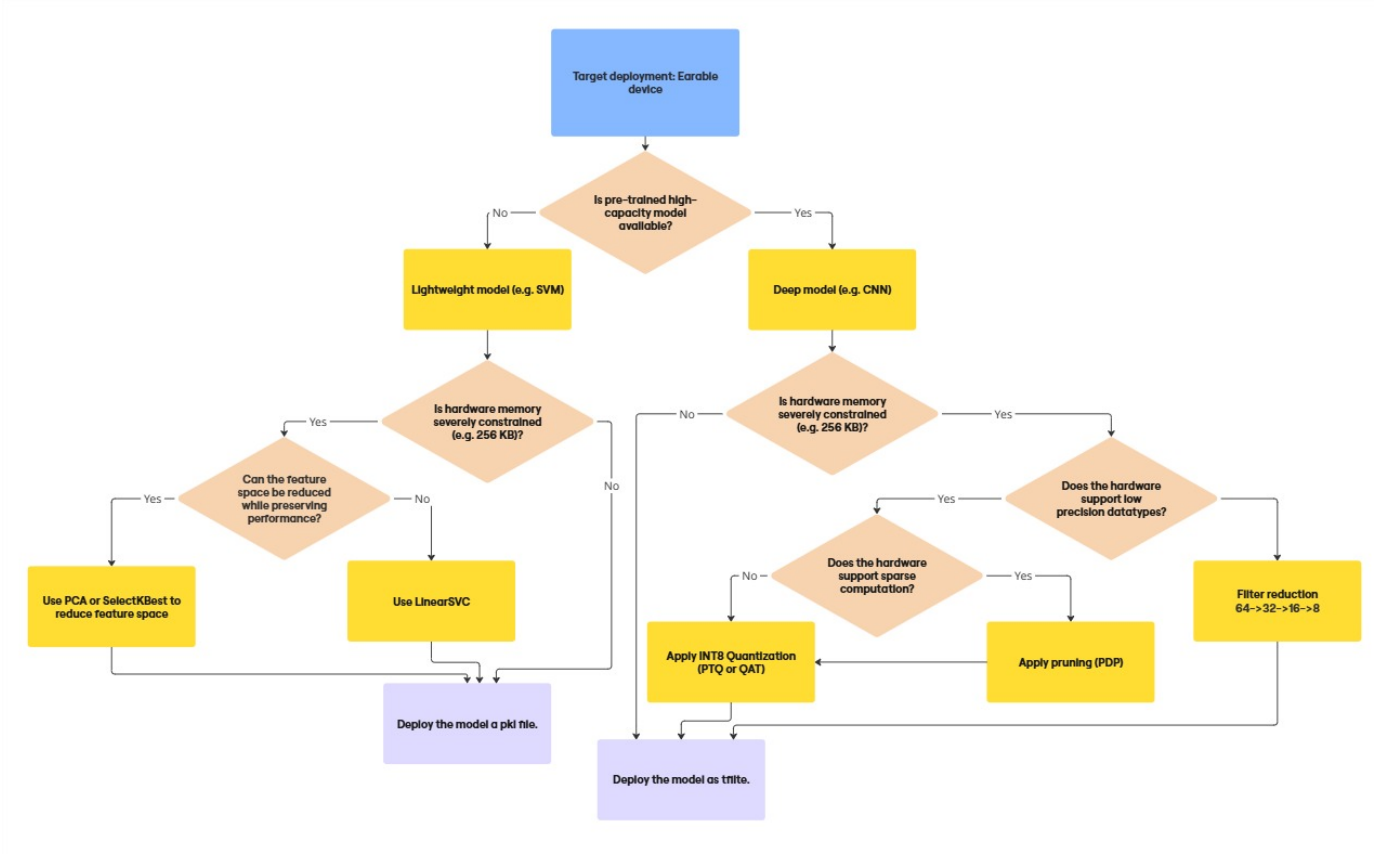


Fig. 12: Model deployment recommendation flowchart

of 0.73 ms but with unsatisfactory performance proving deeper networks cannot perform well when the data volume is limited. TCNs offer a reasonable trade-off, achieving good Kappa score and moderate model size.

Hyperparameters for model tuning are chosen intentionally to maintain model size under 5 MB to ensure the model can be safely deployed onto Raspberry Pi. Hence, the number and depths of trees are limited in Random Forests, the linear kernel is used in SVM and the deep learning models are carefully designed to have fewer layers and parameters.

To address the class imbalance in the dataset, oversampling techniques like SMOTE (Synthetic Minority Over-sampling Technique) are explored for both ML and DL models. However, there was a significant drop in performance (F1 score $< 70\%$). This is because SMOTE synthetically upsamples the data by interpolating between data points, but in IMU time-series data like IMU signals, SMOTE can introduce unrealistic points that can break the natural temporal dependency of the signal. Although, classical shallow models like SVM and Random Forest do not have temporal dependencies, the input features are still extracted from the unrealistic samples of the upsampled data and this may disrupt the statistical patterns that the models learn from. Similarly, undersampling techniques, like reducing samples in the dominant "straight" class, are also avoided to preserve the natural temporal dependencies required for distraction classification.

These findings address **RQ2**, confirming that both machine learning and deep learning approaches can effectively classify visual distraction from earable IMU data. The above comparison of models demonstrates that earable-based IMU features can facilitate moderate and generalizable performance, providing justification for their utilization in real-world cycling scenarios.

C. Deployment results

The analysis of deployment shows some practical issues while optimizing embedded system models. For SVM, although dimensionality reduction helped to minimize model size, further attempts to minimize complexity through hyperparameter optimization like reducing the regularization parameter C to reduce the number of support vectors are explored. While this reduces model size in theory, it resulted in a noticeable reduction in classification performance and thus was not explored further. LinearSVC kernel can also be combined with feature space reduction techniques to reduce the model size even further, but it is not explored as the current LinearSVC variant produced desirable results. Additionally, if models like Random Forest were under consideration, other compression techniques like pruning of trees, feature subsampling, or model distillation would have been available to be

applied to reduce compute complexity and memory footprint without sacrificing classification performance.

For CNNs, several deployment-oriented strategies are explored. Filter reduction substantially decreased model size and inference time, which is particularly useful for deployment on microcontrollers without support for low-precision hardware (e.g., Raspberry Pi Pico or STM32F0). Aggressive filter reduction did cause some significant drops in classification accuracy, indicating the value of fine-tuning. Post-training INT8 quantization (PTQ) and quantization-aware training (QAT) both proved to be very effective. Though both lowered the model size to 0.19 MB and achieved sub-millisecond inference times, QAT surprisingly outperformed the baseline in classification accuracy by nearly 1%, uncommon given QAT typically desires to preserve, not improve, accuracy. This unexpected behavior may be attributed to QAT quantization noise noise acting as a regularizer, helping the model generalize better and reducing overfitting compared to the baseline.

A polynomial decay pruning (PDP) schedule is used to gradually increase sparsity level from 0% to 50% during training in such a way that the model is able to adapt more smoothly compared to constant pruning (CP), which applies fixed sparsity level abruptly. The smooth transition allows the model to learn from the removal of low-magnitude weights without a sudden performance drop, thus making pruning more stable and efficient. The pruning schedule starts after a few initial epochs and continues until a particular training step, so that the model learns a sparse but expressive representation.

To further explore quantization behavior, layer-wise quantization sensitivity and progressive quantization tests are run using TensorFlow's QuantizeWrapper from the TensorFlow Model Optimization Toolkit (TFMOT). This is done by substituting a single layer with its quantized version, while the rest of the model remains FLOAT32. This results in extremely low accuracy. Because non-quantized layers are not rescaled, the activations and internal distributions are not rebalanced, which causes instability and poor predictions. These findings underscore the importance that proper quantization needs to be done in an overall manner throughout the model and not layer-wise in isolation.

These outcomes respond to **RQ3**. Through the application of quantization and model compression techniques, both the CNN and SVM models were compressed into under 256 kilobytes and achieved sub-millisecond inference latency. This proves their effectiveness for deployment on edge devices such as the Arduino Nano 33 BLE Sense.

D. Limitations and future work

This paper presents a feasible detection system for visual distraction using IMU integrated earables. But there are several limitations which pave the way for future implications.

The dataset created is sufficient for initial evaluation. The dataset was collected in semi-controlled cycling conditions and does not involve complex outdoor settings, including environmental factors, lane crossing and interaction with other parties. Future research should aim to collect a more diverse dataset across different terrains, traffic scenarios and weather conditions.

Although head movement are used as a proxy for categorizing visual distraction, it introduces certain limitations. While the earables provide an unobtrusive and low-cost replacement for the eye-tracking glasses, the study assumes a direct correlation between head movement and visual attention. Not all head movements constitute visual distraction. Cyclists, for instance, may shift their heads while keeping their eyes on the road, leading to false positives. These instances are not labeled as distractions in the dataset, but in actual deployment, the system may still classify them as such. Similarly, certain distractions may occur without any head movement especially when cyclist avert their gaze from the road, giving rise to false negatives.

The participant pool is limited in size (20 participants) and demographic diversity (young participants between the ages of 20 and 27). The distraction behaviors and head movements vary a lot with age, physical ability and riding experience. Individuals with visual and hearing impairments may have frequent head movements as a compensatory behaviour, and these head motions may be classified as distractions. Therefore, expanding the demographic coverage is important to improve model generalization.

Manual synchronization is carried out to simultaneously start the video recording and IMU recording. This may introduce some temporal misalignment/delays in the collected data. Furthermore, during data analysis and labelling, packet loss, sensor drift and missing data points are observed between the sessions, likely due to the fact that the earable is powered by a low-power microcontroller. While the observed sensor drift was small, it raises concerns for a long-term deployment as cumulative drifts in acceleration and gyroscope data could introduce errors that degrade model performance. Future scope could involve implementing drift correction algorithms, such as stronger calibration and sensor fusion methods.

Oversampling techniques like SMOTE were found to degrade the performance of the model. Future work should focus on investigating other methods for handling imbalances. Additionally, to address the performance issues on heavy deep learning models like CNN-LSTMs and DeepConv-LSTMs, data augmentation techniques (scaling, flipping) can be explored to improve model generalization.

For the deployment, the optimized models are evaluated on a Raspberry Pi 4; however, the end target is the OpenEarable powered by the Arduino Nano 33 BLE Sense and has much stricter memory and computational constraints. Due to time constraints, real-time deployment on the Arduino is not conducted. Rather, the Raspberry Pi is used for evaluating the feasibility of deployment in terms of model size, inference time, and hardware compatibility. Although unstructured pruning is able to remove low-magnitude weights and reduce active parameters, it does not lead to the reduction in file size since the underlying tensor dimensions are not changed. Additionally, since sparse matrix operations are not supported on both the Raspberry Pi and Arduino Nano 33, the inference time does not become appreciably smaller compared to the

baseline. Moreover, advanced model compression techniques such as knowledge distillation, where a tiny model is trained to mimic a large one, and neural architecture search (NAS), which automatically explores optimal network designs under deployment constraints, are not explored in this study but represent promising directions for future work. These techniques may further reduce model size and computation and, therefore, be especially valuable when deploying models on hardware like the OpenEarables.

VI. Ethical considerations

The models proposed in this research accurately detected visual distractions in most scenarios. However, it is important to consider the ethical limitations and responsible usage of the models. The models are trained and tested on a limited dataset of only 20 participants. This limits the models from being generalized to a larger population across demographic groups such as gender, age, ethnicity and geographical background. Thus, applying the models to a broader context leads to biased or inaccurate outcomes.

Moreover, the system has been implemented as an experimental research prototype. It is not intended for high-stakes decision-making or surveillance. Its application to real-life scenarios without contextual validation can trigger false positives or misclassifications. It presents third-party misuse risks for organizations such as insurance companies or regulatory bodies that can utilize model predictions outside the context, leading to discriminatory or biased procedures.

While the model is limited to detecting visual distractions and does not account for cognitive or auditory factors, it still serves as a step toward broader cyclist distraction detection. To prevent potential abuse, it is necessary that any use of this system is preceded by clear disclaimers, applied solely within its limited research scope, and integrated into broader safety and surveillance infrastructures that involve human oversight. Future iterations of this study should also address model fairness, employ larger and more diverse datasets, and provide open documentation of system limitations to ensure ethical integrity and protect user rights.

VII. Conclusion

Visual distraction detection, especially in the realm of cycling, remains underexplored. This paper explores a novel approach to detecting visual distraction in cyclists using IMU-based earable sensors collected from head movement data. Unlike relying on surveys, datasets and expensive equipment like virtual environments, the proposed work makes notable contributions by leveraging a lightweight ear-worn sensor, which enables naturalistic monitoring.

A custom dataset is created for recording head movements indicative of visual distraction behaviors during cycling scenarios. The dataset addresses the lack of open-source data focusing on dynamic, outdoor scenarios with earables. The dataset comprises recordings from 20 participants, capturing natural head movement patterns during both distracted and non-distracted events. It provides a foundation for developing visual distraction systems and can be extended for future work on mobility safety applications.

Machine learning models (SVM, Random Forest, XgBoost, KNN) and deep learning models (CNN,CNN-LSTM, DeepConv-LSTM, TCN) are trained using the dataset. To ensure feasible deployment onto an edge device, models are designed to have the best performance where the model size is below 5 MB.

The results suggest that visual distractions among cyclists can be moderately classified using IMU data from head movements. While SVM and CNN models offer the best trade-off in F1-score, chance-corrected measure (Kappa score) and memory footprint, the overall performance shows that there is still room for improvement, particularly in identifying minority distraction classes.

The deployment analysis shows that with the right optimization techniques, both classical and deep learning models can be adapted for devices with limited resources. For SVM deployment, LinearSVC stands out as the lightweight option that supports high-speed inference with minimal performance loss. INT8 PTQ and INT8 QAT variants reduce the model size with little to no performance loss in the case of CNN deployment. Although deployment experiments run on the Raspberry Pi 4, they serve as a good proxy for embedded systems feasibility evaluation. The proposed systematic deployment approach of feature space reduction, pruning, quantization, and filter reduction offers a usable guide for mapping accurate models onto runtime wearable applications using the OpenEarable platform.

Overall, this work establishes a comprehensive foundation for real-world distraction monitoring using earable sensors, covering the complete pipeline from dataset creation and model training to evaluating deployment feasibility on resource-constrained wearable platforms.

Acknowledgment

The author would like to thank Özlem Durmaz and Akhil Pallamreddy for their guidance and support throughout this research. During the preparation of this work, the author utilized AI tools for assistance with debugging and refining the report. All content was thoroughly reviewed and edited as needed.

References

[1] E. H. Rad, F. Kavandi, L. Kouchakinejad-Eramsadati, K. Asadi, and N. Khodadadi-Hassankiadeh, "Self-reported cycling behavior and previous history of traffic accidents of cyclists," *BMC Public Health*, vol. 24, no. 1, 2024.

[2] SWOV Institute for Road Safety Research, "Distraction in traffic," https://swov.nl/en/fact-sheet/distraction-traffic, n.d., accessed: 2024-12-22.

[3] J. S. Mindell, D. Leslie, and M. Wardlaw, "Exposure-based, 'like-for-like' assessment of road safety by travel mode using routine health data," *PLoS ONE*, vol. 7, no. 12, p. e50606, 2012.

[4] SWOV Institute for Road Safety Research, "Cyclists," https://swov.nl/en/fact-sheet/cyclists, n.d., accessed: 2024-12-22.

[5] D. De Waard, B. Lewis-Evans, B. Jelijs, O. Tucha, and K. Brookhuis, "The effects of operating a touch screen smartphone and other common activities performed while bicycling on cycling behaviour," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 22, p. 196–206, 2014.

[6] X. Guo, A. Tavakoli, T. D. Chen, and A. Heydarian, "Unveiling the impact of cognitive distraction on cyclists psycho-behavioral responses in an immersive virtual environment," *IEEE Transactions on Intelligent Transportation Systems*, p. 1–12, 2024.

[7] M. A. Regan, C. Hallett, and C. P. Gordon, "Driver distraction and driver inattention: Definition, relationship and taxonomy," *Accident Analysis & Prevention*, vol. 43, no. 5, pp. 1771–1781, 2011.

[8] D. Ethan, C. H. Basch, G. D. Johnson, R. Hammond, C. M. Chow, and V. Varsos, "An analysis of technology-related distracted biking behaviors and helmet use among cyclists in new york city," *Journal of Community Health*, vol. 41, no. 1, p. 138–145, 2015.

[9] K. L. Young, A. N. Stephens, S. O'Hern, and S. Koppel, "Australian cyclists' engagement in secondary tasks," *Journal of Transport & Health*, vol. 16, p. 100793, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214140519303202

[10] S. A. Useche, F. Alonso, L. Montoro, and C. Esteban, "Distraction of cyclists: how does it influence their risky behaviors and traffic crashes?" *PeerJ*, vol. 6, p. e5616, 2018.

[11] G. d. Smit, D. Yeleshetty, P. J. Havinga, and Y. Huang, "Predicting turn maneuvers of cyclists using bicycle-mounted imu with cnn-lstm," in *2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2024, pp. 587–592.

[12] Z. Han, L. Xu, X. Dong, Y. Nishiyama, and K. Sezaki, "Headmon: Head dynamics enabled riding maneuver prediction," in *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2023, pp. 22–31.

[13] S. Zhang and M. Abdel-Aty, "Drivers' visual distraction detection using facial landmarks and head pose," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2676, p. 036119812210872, 04 2022.

[14] T. Röddiger, C. Clarke, P. Breitling, T. Schneegans, H. Zhao, H. Gellersen, and M. Beigl, "Sensing with earables: A systematic literature review and taxonomy of phenomena," vol. 6, no. 3, 2022. [Online]. Available: https://doi.org/10.1145/3550314

[15] F. Kawsar, C. Min, A. Mathur, and A. Montanari, "Earables for personal-scale behavior analytics," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 83–89, 2018.

[16] T. Röddiger, T. King, D. R. Roodt, C. Clarke, and M. Beigl, "Openearable: Open hardware earable sensing platform," in *Proceedings of the 1st International Workshop on Earable Computing*, ser. EarComp'22. New York, NY, USA: Association for Computing Machinery, 2023, pp. 29–34. [Online]. Available: https://doi.org/10.1145/3544793.3563415

[17] R. Oyini Mbouna, S. G. Kong, and M.-G. Chun, "Visual analysis of eye state and head pose for driver alertness monitoring," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1462–1469, 2013.

[18] L. Alam and M. M. Hoque, "Real-time distraction detection based on driver's visual features," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–6.

[19] S. Martin, S. Vora, K. Yuen, and M. M. Trivedi, "Dynamics of driver's gaze: Explorations in behavior modeling & maneuver prediction," 2018. [Online]. Available: https://arxiv.org/abs/1802.00066

[20] A. Doshi and M. M. Trivedi, "On the roles of eye gaze and head dynamics in predicting driver's intent to change lanes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 453–462, 2009.

[21] Y. Gu, Z. Shao, L. Qin, W. Lu, and M. Li, "A deep learning framework for cycling maneuvers classification," *IEEE Access*, vol. 7, pp. 28 799–28 809, 2019.

[22] F. Vicente, Z. Huang, X. Xiong, F. De la Torre, W. Zhang, and D. Levi, "Driver gaze tracking and eyes off the road detection system," *Trans. Intell. Transport. Syst.*, vol. 16, no. 4, p. 2014–2027, Jul. 2015. [Online]. Available: https://doi.org/10.1109/TITS.2015.2396031

[23] Z. Han, X. Dong, Y. Nishiyama, and K. Sezaki, "Headsense: A head movement detecting system for micro-mobility riders," 09 2021, pp. 26–27.

[24] K. Wong, Y.-C. Chen, T.-C. Lee, and S.-M. Wang, "Head motion recognition using a smart helmet for motorcycle riders," in *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, 2019, pp. 1–7.

[25] Y.-R. Chen, C.-M. Tsai, K.-I. Wong, T.-C. Lee, C.-H. Loh, J.-C. Ying, and Y.-C. Chen, "Motorcyclists' head motions recognition by using the smart helmet with low sampling rate," in *2019 Twelfth International Conference on Ubi-Media Computing (Ubi-Media)*, 2019, pp. 157–163.

[26] Y. Usami, K. Ishikawa, T. Takayama, M. Yanagisawa, and N. Togawa, "Bicycle behavior recognition using sensors equipped with smartphone," in *2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, 2018, pp. 1–6.

[27] W. Gu, Y. Liu, Y. Zhou, Z. Zhou, C. J. Spanos, and L. Zhang, "Bikesafe: bicycle behavior monitoring via smartphones," ser. UbiComp '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 45–48. [Online]. Available: https://doi.org/10.1145/3123024.3123158

[28] W. Gu, Z. Zhou, Y. Zhou, H. Zou, Y. Liu, C. J. Spanos, and L. Zhang, "Bikemate: Bike riding behavior monitoring with smartphones," in *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ser. MobiQuitous 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 313–322. [Online]. Available: https://doi.org/10.1145/3144457.3144462

[29] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava, "Biketastic: sensing and mapping for better biking," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1817–1820. [Online]. Available: https://doi.org/10.1145/1753326.1753598

[30] L. Jiang, X. Lin, X. Liu, C. Bi, and G. Xing, "Safedrive: Detecting distracted driving behaviors using wrist-worn devices," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 4, Jan. 2018. [Online]. Available: https://doi.org/10.1145/3161179

[31] L. Shojaeifard, A. Islam, H. Shaheen, V. Schroderus, and E. Peltonen, "Left or right? detecting driver's head movement on the road," 11 2023.

[32] T. Sepanosian and O. Durmaz Incel, "Boxing gesture recognition in real-time using earable imus," in *Companion of the 2024 on ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 673–678. [Online]. Available: https://doi.org/10.1145/3675094.3680524

[33] M. Radhakrishnan, K. Misra, and V. Ravichandran, "Applying "earable" inertial sensing for real-time head posture detection," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2021, pp. 176–181.

[34] M. Laporte, P. Baglat, S. Gashi, M. Gjoreski, S. Santini, and M. Langheinrich, "Detecting verbal and non-verbal gestures using earables," in *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*, ser. UbiComp/ISWC '21 Adjunct. New York, NY, USA: Association for Computing Machinery, 2021, p. 165–170. [Online]. Available: https://doi.org/10.1145/3460418.3479322

[35] S. Gashi, A. Saeed, A. Vicini, E. Di Lascio, and S. Santini, "Hierarchical classification and transfer learning to recognize head gestures and facial expressions using earbuds," in *Proceedings of the 2021 International Conference on Multimodal Interaction*, ser. ICMI '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 168–176. [Online]. Available: https://doi.org/10.1145/3462244.3479921

[36] T. Hossain, M. S. Islam, M. A. R. Ahad, and S. Inoue, "Human activity recognition using earable device," in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, ser. UbiComp/ISWC '19 Adjunct. New York, NY, USA: Association for Computing Machinery, 2019, p. 81–84. [Online]. Available: https://doi.org/10.1145/3341162.3343822

[37] A. Ferlini, A. Montanari, C. Mascolo, and R. Harle, "Head motion tracking through in-ear wearables," in *Proceedings of the 1st International Workshop on Earable Computing*, ser. EarComp'19. New York, NY, USA: Association for Computing Machinery, 2020, p. 8–13. [Online]. Available: https://doi.org/10.1145/3345615.3361131

[38] C.-W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de Oca, Y. Cheng, M. Lin, L. Torresani, and A. T. Campbell, "Carsafe app: alerting drowsy and distracted drivers using dual cameras on smartphones," ser. MobiSys '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 13–26. [Online]. Available: https://doi.org/10.1145/2462456.2465428

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Van-

derplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[40] F. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.
[41] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data." *Biometrics*, vol. 33 1, pp. 159–74, 1977. [Online]. Available: https://api.semanticscholar.org/CorpusID:11077516
[42] N. Sheikh, "On the design of efficient deep learning methods for human activity recognition in resource constrained devices," Master's Thesis, University of Waterloo, Waterloo, Ontario, Canada, 2023.

APPENDIX A: PHONE CALL QUESTIONS

The questions asked to the participants during the phone call session (Session 5) are as follows:

- What did you eat for your breakfast? Can you describe it in detail, including ingredients used?
- Tell me about your morning routine from the moment you woke up until you left home.
- What were the last three items you purchased, and why did you choose them?
- Can you remember your favourite childhood game? Describe how it was played.
- Describe the last event or celebration you attended. What made it special?
- If you could invent a new gadget to make life easier, what would it be and how would it work?
- Imagine you won the lottery—what's the first thing you would do?
- If you could time travel to any era, where would you go and why?
- If you could have dinner with any three famous people, who would they be and why?
- If you could time travel to any era, where would you go and why?
- What superpower would you like to have? and why?