applied sciences

MDPI

Article

Pre-Impact Fall Detection for E-Scooter Riding Using an IMU: Threshold-Based, Supervised, and Unsupervised Approaches

Seunghee Lee, Bummo Koo and Youngho Kim *

Department of Biomedical Engineering, Yonsei University, Wonju 26493, Republic of Korea; fhrm502@yonsei.ac.kr (S.L.); beommo@yonsei.ac.kr (B.K.)
* Correspondence: younghokim@yonsei.ac.kr; Tel.: +82-33-760-2859

Abstract: Pre-impact fall detection during e-scooter riding is essential for rider safety. Both threshold-based and deep learning algorithms (supervised and unsupervised models) were developed in this study. Twenty participants performed normal driving maneuvers such as straight driving, speed bumps, clockwise roundabouts, and counterclockwise roundabouts, along with falls (abnormal driving maneuvers). A 6-axis IMU sensor (Xsens DOT, The Netherlands) was positioned at the T7 location to record data at 60 Hz. The approaches included threshold-based, supervised learning, and unsupervised learning models The threshold-based approach yielded an accuracy of 98.86% with an F1 score of 0.99, while the supervised model had a slightly lower performance, reaching 86.29% accuracy and an F1 score of 0.56. The unsupervised knowledge distillation model achieved 98.86% accuracy, an F1 score of 0.99, and a memory size of only 46 kB. All models demonstrated lead times of more than 250 ms, sufficient for airbag deployment.

Keywords: pre-impact fall detection; electric scooter; unsupervised model; autoencoder; lightweight

Citation: Lee, S.; Koo, B.; Kim, Y. Pre-Impact Fall Detection for E-Scooter Riding Using an IMU: Threshold-Based, Supervised, and Unsupervised Approaches. *Appl. Sci.* **2024**, *14*, 10443. https://doi.org/10.3390/app142210443

Academic Editors: Ping Wang and Linlin You

Received: 26 September 2024
Revised: 1 November 2024
Accepted: 12 November 2024
Published: 13 November 2024

1. Introduction

Electric scooters (e-scooters) have recently gained popularity as a personal means of transportation due to their affordability and ease of use. Additionally, their eco-friendly nature results in low carbon emissions, making them a potential alternative to bicycles and traditional scooters [1,2]. Thanks to these advantages, the e-scooter market has rapidly expanded and now covers 626 cities across 53 countries [3]. According to a report by Nimble app Genie [4], the global e-scooter market size reached $70.26 billion in 2022, and it is projected to grow at a compound annual growth rate of approximately 9.9% from 2023 to 2030 [5,6].

With the increase in e-scooter usage, related accidents have also surged. According to a report from the Korea Road Traffic Authority, e-scooter-related accidents increased by an average of 96.2% annually from 2017 to 2021, while overall traffic accidents decreased by approximately 1.6% during the same period [7]. Namiri et al. [8] reported that head injury rates for e-scooter users were more than twice as high as those for cyclists, with the primary cause of injuries being falls, leading to head and facial injury rates ranging from 20% to 58%. Posirisuk et al. [9] conducted dynamic simulations to determine impact forces generated during e-scooter falls and reported a significant correlation between the speed and the impact force on the head. They found that the impact forces during falls were great enough to cause fractures.

E-scooter users are vulnerable to injuries due to the lack of protective gear, highlighting the importance of protective equipment like helmets. Wearable sensor-based protective devices have been used to detect falls or crashes and prevent injuries. Ahn et al. [10] developed an algorithm that detected human falls by calculating vertical angles and triangular features using 6-axis inertial sensor data. Their algorithm achieved 100% accuracy with an average lead time (the time difference between the fall and the detection) of over 300 ms

on their own dataset and showed 90.3% accuracy using the Sisfall [11] public dataset. Kim et al. [12] developed a fall-detection algorithm by calculating vertical angles and vertical speeds based on inertial sensor data. The algorithm demonstrated 98.3% accuracy with an average lead time of over 311 ms on their dataset, and it was tested on the KFall [13] and FARSEEING [14] real fall datasets, reporting accuracies of 99.5% and 71.4%, respectively. Kim et al. [15] developed a threshold-based fall-from-height detection algorithm aimed at detecting fall incidents on construction sites. By calculating vertical angles using inertial sensor data and deriving vertical speed, the algorithm achieved 100% accuracy with an average lead time of 240 ms. Abderrahmane et al. [16] detected motorcycle falls using inertial sensors mounted on the vehicle. By applying thresholds to acceleration and angular velocity, they determined fall moments and integrated the system with airbags and other safety devices. Their algorithm was tested with stuntmen simulating fall scenarios at speeds up to 80 km/h, achieving a lead time of over 200 ms.

Recent advances in computer science have led to the integration of AI with wearable sensor research. Yu et al. [17] developed a fall-detection algorithm using a conv-LSTM model, achieving 99.32% accuracy with an average lead time of 403 ms on the KFall public dataset. The algorithm was further validated with the additional data collected from elderly individuals, demonstrating 99.84% accuracy. Koo et al. [18] developed TinyFallNet, a lightweight fall-detection algorithm by tuning the parameters of the ResNet model. The model achieved 98.0% accuracy with an average lead time of over 539 ms and was reduced to 716 kB, achieving approximately a 36% reduction in size. Ramon et al. [19] proposed an algorithm to detect e-scooter falls using onboard inertial sensors. The researchers embedded various machine learning models to detect falls, reporting 97.1% accuracy and an F1 score of 0.970 using a support vector machine model. The anomaly data, generally difficult to collect, are required in order to train classification models for the anomaly detection. Yu et al. [20] tested the performance of a fall-detection model by adjusting the proportion of fall data during training and showed no significant difference in model performance even though only 77% of fall data was used. This suggests that sufficient performance could be achieved even with a small amount of anomaly data.

In this study, falls during e-scooter riding were detected using an IMU sensor before the impact occurred at the ground. Three different approaches were taken for the pre-impact fall detection algorithm: threshold-based, supervised, and unsupervised models. The threshold-based algorithm was proposed using the magnitude of the acceleration, the angular velocity, and the roll angles from the 6-axis IMU data. A supervised learning approach was employed using deep learning models to classify normal and fall scenarios. Finally, an unsupervised learning model was developed using autoencoder models. In addition, knowledge distillation techniques were applied to the unsupervised model in order to reduce the model size, while performance was maintained.

2. Materials and Methods

2.1. Experiment

This study recruited twenty healthy adults (17 males, 3 females; 23.3 $\pm$ 2.0 years old, 173.0 $\pm$ 6.7 cm, 74.5 $\pm$ 13.0 kg) from Yonsei University. Individuals with musculoskeletal problems were excluded from the study. The experiment received approval from the Yonsei University Mirae Campus Institutional Review Board (IRB) (1041849-202109-BM-139-02), and all participants provided written informed consent. An IMU sensor (Xsens Dot, Movella Holdings Inc., Henderson, NV, USA) was attached to the T7 position of each subject. The three-axis acceleration and the three-axis angular velocity signals were measured at a sampling frequency of 60 Hz. The subjects were required to perform four types of normal driving (straight driving, speed-bump driving, counterclockwise (CCW) roundabout driving, and clockwise (CW) roundabout driving) and one type of fall on an e-scooter (Ninebot Segway ES2, Xiaomi Technology Co., Ltd., Beijing, China, Figure 1). They were instructed to drive at a speed of 15 km/h in the straight section and 12 km/h during the turns. The fall involved a fall maneuver, where the subjects were instructed to drive at a

speed of 15 km/h and then perform a fall (Table 1). Participants performed normal driving for a duration of 30 s to 1 min during each trial, covering distances of approximately 20 to 50 m. Data collection started prior to the participants' start and continued until they came to a complete stop. Each normal driving session was recorded at least three times. To balance the dataset with non-fall data, falls were intentionally recorded at least 10 times. The fall was executed by having the front wheel of the scooter make the initial impact, followed by the participant experiencing a second impact with the ground.

Figure 1. The e-scooter and an IMU sensor (position: T7).

Table 1. Experimental movements.

	Code	Description	Speed (km/h)
Normal driving	D1	Straight Driving	15
	D2	Speed-Bump Driving	15
	D3	Counterclockwise Roundabout Driving (CCW)	12
	D4	Clockwise Roundabout Driving (CW)	12
Fall	F1	Fall	15

The real-road surfaces like asphalt are not as smooth and are expected to produce more noise in the sensors. Additional experiments in a more realistic environment were conducted on roads. Only normal driving data were collected, since performing fall maneuvers on roads posed too much danger. Figure 2 illustrates a map of the route for a 3 min trial conducted on general roads. Ten participants repeated the course five times, with each trial consisting of one full lap.

Figure 2. The real-road e-scooter driving.

2.2. Pre-Impact Fall Detection During E-Scooter Riding

2.2.1. Threshold-Based Model

Two sum vector magnitude (SVM) values, acceleration SVM (ASVM), and angular velocity SVM (GSVM) were calculated from the acceleration and angular velocity data measured by an IMU sensor. The x-axis represents the anterior–posterior direction, the y-axis the medial–lateral direction, and the z-axis the vertical direction.

$$ASVM = \sqrt{{Acc_X}^2 + {Acc_Y}^2 + {Acc_Z}^2} \quad (1)$$

$$GSVM = \sqrt{{Gyro_X}^2 + {Gyro_Y}^2 + {Gyro_Z}^2} \quad (2)$$

To determine the roll angle, a complementary filter was used, and quaternion operations were applied to solve the gimbal lock problem. To reduce noise, a Butterworth filter was employed with the following specifications: a 2nd-order low-pass filter with a cutoff frequency of 5 Hz. The filter was applied to data sampled at a rate of 60 Hz.

Figure 3 illustrates the flowchart of the threshold-based pre-impact fall-detection algorithm for e-scooters. In this context, A_{Th}, G_{Th}, and $Angle_{Th}$ represent the thresholds for $ASVM$, $GSVM$, and roll angle, respectively. A fall was detected when both the acceleration and angular velocity thresholds were satisfied, followed by the angle threshold being met within 0.2 s. The frame satisfying the angle threshold was considered the fall-detection frame. Additional filtering on the angle helps to filter out false positives that might occur during normal riding conditions. Due to the characteristics of e-scooters, which often experience medial–lateral angle changes, the algorithm only considered the roll angle, which related to the anterior–posterior direction.

A_{Th} was set in the range of 5 g to 20 g at 1 g intervals, G_{Th} was set in the range of 50 deg/s to 300 deg/s at 10 deg/s intervals, and $Angle_{Th}$ was set in the range of 20 deg to 50 deg at 5 deg intervals to determine the optimal thresholds using grid search. The thresholds were determined using data from 16 out of 24 participants, while the evaluation of the algorithm was performed using data from the remaining 8 participants. Accuracy, sensitivity, precision, F1 score, and lead time were calculated as follows:

$$Accuracy\ (\%) = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (3)$$

$$Sensitivity\ (\%) = \frac{TP}{TP + FN} \times 100 \quad (4)$$

$$Precision\ (\%) = \frac{TP}{TP + FP} \times 100 \tag{5}$$

$$F1\ Score = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} \tag{6}$$

$$Lead - time\ (ms) = \frac{Impact\ frame - Detected\ frame}{Sampling\ rate} \times 1000 \tag{7}$$

where *TP*, *FP*, *TN*, and *FN* represent true positives, false positives, true negatives, and false negatives, respectively. *TP* is the number of falls detected as falls, *FP* is the number of normal driving instances detected as falls, *TN* is the number of normal driving instances detected as normal driving, and *FN* is the number of falls detected as normal driving. The lead time is the period between the frame when impact occurs at the ground during the fall and the frame when the fall was detected. A frame refers to a single IMU data measured at a frequency of 60 Hz.

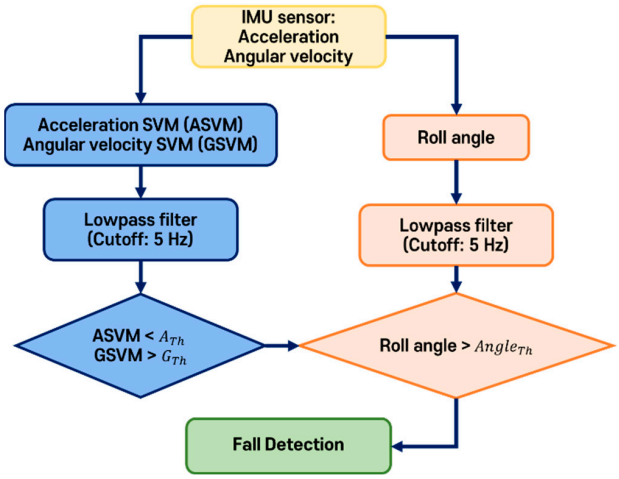

Figure 3. Threshold-based pre-impact fall-detection algorithm flow chart.

2.2.2. Supervised Model

A sliding-window technique was applied to prepare input data for the supervised deep learning model, with a window size of 1 s and a 20% overlap [18]. All normal riding movements were labeled as non-fall. For fall movements, the period started from 0.2 s before ground impact was labeled as the fall onset. If a window contained more than 20% of fall onset data, the window was labeled as a fall. For model training, the data were split into a 2:1 ratio for training and testing, with 20% of the training data set aside for validation during the training process. The non-fall data were undersampled to match the number of fall windows due to the significant imbalance between the two. The number of fall windows used for training was about 400, while the number of non-fall windows was approximately 9000. To adjust the ratio between the two datasets, 320 non-fall windows were randomly selected. The slightly higher number of fall windows was intended to emphasize falls during training.

The model was built based on the ResNet-18 architecture, a residual network that consists of 18 layers and leverages skip connections to overcome the vanishing gradient problem, allowing for more efficient training of deep networks. The ResNet model is typically used for image-related problems, but it has recently demonstrated strong performance in the time series domain as well [18]. The input data size for the model was determined to be 6 × 60, based on the 6-axis inertial sensor data and the window size. This architecture is well suited for extracting complex features from data while maintaining a relatively lightweight model. For training and testing, the softmax function was used, and windows with a probability value of 0.65 or higher were classified as falls [17]. To prevent overfitting, a max-pooling layer was applied to the ResNet-18 model, and early

stopping was applied during training with a validation ratio of 20%, stopping the process if validation accuracy did not improve for 20 consecutive epochs. To enhance the algorithm's robustness, a fall was confirmed when two consecutive fall windows were detected. The last frame of the second fall window was determined as the fall detection frame, and the lead time was calculated as the difference between this frame and the impact frame. Accuracy, sensitivity, precision, and F1 score were calculated following the same approach as described in Equations (3)–(7).

2.2.3. Unsupervised Model

The same sliding-window technique was applied as in the supervised model (Section 2.2.1). The window size was 1 s, corresponding to 60 frames, with a 50% overlap [21]. The input data consisted of 6 features: 3 axes of acceleration and 3 axes of angular velocity, resulting in a data dimension of 6×60. The data were divided into training and test datasets with a 2:1 ratio, and 20% of the training dataset was used for model validation during training. Normal driving data were used as the normal data, while fall data were considered abnormal. Only the normal data were used for training the model.

Knowledge distillation is a technique used to transfer knowledge from a large, complex model, known as the "teacher", to a smaller, more efficient model, referred to as the "student" [22]. The goal is to compress the teacher's knowledge into the student model while preserving its performance and generalization capabilities. The teacher model consisted of a 3-layer CNN encoder with filter sizes of 128, 64, and 64, respectively. LeakyReLU activation functions were applied in each layer, and max pooling was used between layers to prevent overfitting. The decoder also employed a 3-layer CNN with filter sizes of 64, 64, and 128, respectively, and LeakyReLU activation functions were applied. The student model comprised a 3-layer CNN encoder and a 3-layer decoder, with filter sizes of 8, 8, and 4 in the encoder, and 4, 8, and 8 in the decoder, respectively. Similar to the teacher model, LeakyReLU activation functions were used for both the encoder and decoder. Max pooling was applied in the encoder to prevent overfitting.

Reconstruction error [23] refers to the difference between the input and the output data of an autoencoder and was calculated as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{8}$$

where x_i represents the original value of the input window, and $\hat{x}_i$ represents the value of the reconstructed output window.

The autoencoder model (Figure 4) was trained only for the normal data from the training dataset, and the maximum reconstruction error observed during training was set as the threshold. Any window data exceeding this threshold were considered anomalous. If two consecutive windows were classified as anomalous, the end frame of the second window was designated as the detected frame, and the data were then classified as a fall. Accuracy, sensitivity, precision, F1 score, and lead time were calculated as Equations (3)–(7).

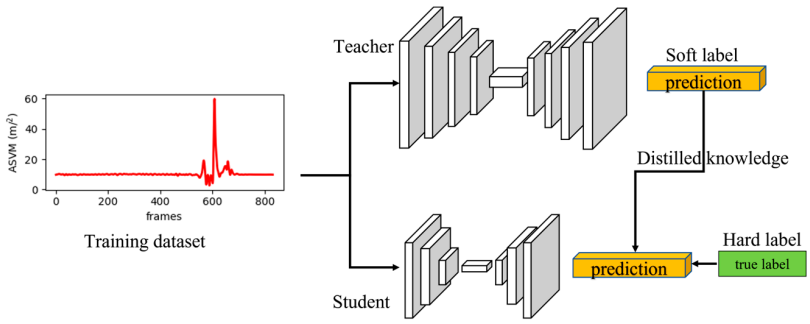

Figure 4. Deep learning autoencoder model.

3. Results

Table 2 summarizes the performance metrics of the various models evaluated in this study, including the threshold-based model, supervised model, and unsupervised models (teacher model, student model, and knowledge distillation (KD) model). The threshold-based model demonstrated an accuracy of 98.86%, indicating its robustness in fall detection. The supervised model achieved an accuracy of 86.29%, showing reasonable performance but less effectiveness compared to the other models. The unsupervised teacher model from the unsupervised approach achieved the highest accuracy of 99.43%, reflecting its superiority in detecting falls. The unsupervised student model, though slightly lower, maintained an accuracy of 92.22%, while the KD model closely followed with an accuracy of 98.86%, demonstrating its ability to maintain strong performance through knowledge transfer. In terms of the sensitivity, which reflects the model's ability to detect falls correctly, the unsupervised teacher model performed well with a sensitivity of 100%, highlighting its effectiveness in identifying all fall events. The threshold-based model performed well with a sensitivity of 97.89%, while the KD model also performed strongly, matching the unsupervised teacher model at 100%. The supervised and unsupervised student models showed relatively lower sensitivities of 88.89% and 88.10%, respectively. Precision, reflecting the accuracy of positive fall predictions, was highest for the threshold-based model at 100%, followed closely by the KD model at 97.93%. The unsupervised teacher model demonstrated a reliable performance with a precision of 98.91%, while the unsupervised student model achieved a precision of 96.10%. The supervised model had a precision of 82.76%. The F1 score, balancing sensitivity and precision, was also high for all models, with the unsupervised teacher model and KD model both achieving scores of 0.99, followed by the threshold-based model at 0.99. The supervised model had an F1 score of 0.86, while the unsupervised student model had a slightly higher F1 score of 0.92. The threshold-based model had the longest lead time at 665.05 $\pm$ 212.5 ms. The supervised model demonstrated the shortest lead time of 275.46 $\pm$ 172.87 ms. Among the unsupervised models, the unsupervised teacher model had a lead time of 495.42 $\pm$ 208.50 ms, followed by the KD model's of 457.1 $\pm$ 251.76 ms and the unsupervised student model's of 402.2 $\pm$ 297.91 ms. The KD model demonstrated the smallest memory size of 46 kB.

Table 2. Classification performances of models.

	Threshold-Based	**Supervised**	**Unsupervised**		
			Teacher	**Student**	**KD**
Accuracy (%)	98.86	86.29	99.43	92.22	98.86
Sensitivity (%)	97.89	88.89	100	88.10	100
Precision (%)	100	82.76	98.95	96.10	97.93
F1 score	0.99	0.56	0.99	0.92	0.99
Lead time (ms)	665.05 $\pm$ 212.5	275.46 $\pm$ 172.87	495.42 $\pm$ 208.50	402.2 $\pm$ 297.91	457.1 $\pm$ 251.76
Memory size (kB)	-	3785	1437	85	46

Figure 5 illustrates the confusion matrices for different models. A total of 175 samples were used in the test dataset, consisting of 80 non-fall data points and 95 fall data points. The threshold-based model showed false detections in two fall movements. For the supervised model, a total of 15 movements were classified as false positives, which included 1 normal driving instance being incorrectly detected as a fall and 14 fall movements being detected too late. Due to the early detection of a fall, 1 and 2 false positives were observed in the unsupervised teacher and the KD models, respectively. Early detection refers to instances where a fall is not detected in a timely manner, resulting in a lead time of more than 3 s.

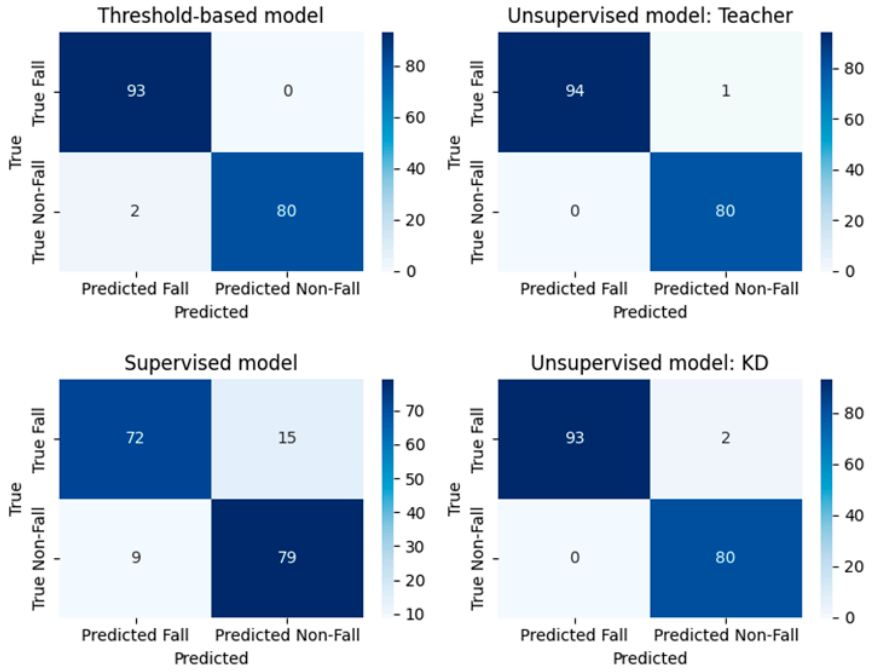

Figure 5. Confusion matrices in different models.

Table 3 presents the results from the real-road trials. All models showed 100% true negatives (TN) in 50 trials. Normal driving should be classified as TN when all windows are classified as non-fall in a single trial.

Table 3. Model performance on real-road driving data.

	Threshold-Based	Supervised	Unsupervised		
			Teacher	Student	KD
TN	50	50	50	50	50

4. Discussion

In this study, falls during e-scooter riding were detected using an IMU sensor before the impact occurred at the ground. Threshold-based, supervised, and unsupervised models were developed. In addition, the lightweighting of the model was achieved using knowledge distillation in the unsupervised learning approach.

The threshold-based model achieved an accuracy of 98.86%, effectively distinguishing between falls and normal driving, with an F1 score of 0.99. Two false negatives in the threshold-based model occurred likely due to the foot touching the ground before the fall, which delayed the change in angular velocity; however, these two FNs were successfully detected by unsupervised learning-based teacher and KD models. The threshold-based model also exhibited the longest average lead time of 665.05 ms. It is expected to use less computational memory compared to AI models. In the case of the supervised model, the performance was somewhat lower, with an accuracy of 86.29%, mainly due to the imbalance between the normal and the fall data. Although undersampling techniques were applied to minimize this imbalance, the number of fall data samples was still limited. Therefore, it will be essential to collect more fall data in future work to improve training and model performance. The unsupervised learning-based teacher and KD models both recorded an accuracy of over 98%, successfully detecting falls. The false-positive errors in the unsupervised learning teacher and KD models occurred because of the early detections. Although their lead times were slightly shorter than that of the threshold-based model, they still provided sufficient time for deploying protective devices such as airbags [24].

In the case of normal driving in the real world, all models demonstrated 100% specificity, showing good performance despite the potential sensor noise that may occur on the road surface. The algorithms developed in this study are expected to be applied to protective devices, such as wearable airbags, to enhance the safety of riders.

Ramon et al. [19] reported an accuracy of 97.1% and an F1 score of 0.970 in their study on fall detection using inertial sensors attached to an e-scooter. They employed TinyML to integrate machine learning models into edge devices, focusing on aspects like computation time. However, their study did not address lead time, which is crucial for detecting falls before impact. Attal et al. [25] developed a fall detection algorithm based on IMU embedded in motorcycles and validated it through 29 scenario movements. They reported a lead time of over 260 ms, which is sufficiently longer than the 100 ms required for rider jacket deployment. In this study, the lead time of all models was sufficiently long.

The KD model successfully reduced the memory size by approximately 96%, compared to the unsupervised teacher model, down to 46 kB. The memory sizes in Table 2 were expressed in the .h5 Keras format, but can be further reduced when converted to TensorFlow Lite (.tflite). Thus, owing to the knowledge distillation technique, a lightweight model was created while maintaining a certain level of performance of the teacher model, possibly making it suitable for application on edge devices such as smartphones.

This study has several limitations. It is extremely dangerous to measure actual falls or to induce them; thus, fall data are generally collected artificially. Another limitation is that most participants were in their early twenties, which makes it difficult to generalize the findings to other age groups. E-scooters are generally used by young people, and the nature of fall movements during e-scooter riding does not differ in the young and the elderly. Lastly, the sensor was placed at the T7 location, specifically on the back. However, the signal patterns are likely to differ if the sensor is positioned elsewhere, such as on a wristwatch or in a smartphone in a pocket.

5. Conclusions

In this study, pre-impact fall-detection algorithms for e-scooter riding were developed using an IMU sensor positioned at the T7 location. Three different approaches were applied: threshold-based, supervised learning, and unsupervised learning. The threshold model demonstrated excellent performance with an accuracy of 98.86%. The supervised learning model, however, showed a relatively lower accuracy of 86.29%, which was attributed to the imbalance between normal driving data and fall data. An anomaly detection model based on unsupervised learning was developed to address the data imbalance, resulting in an accuracy of 98.86%. The application of knowledge distillation techniques to this unsupervised model significantly reduced the model size by approximately 96%. As a result, unsupervised learning effectively handled the problem of data imbalance without requiring extensive collection of abnormal data.

Author Contributions: Conceptualization, S.L., B.K. and Y.K.; methodology, S.L.; software, S.L.; validation, B.K. and Y.K.; investigation, B.K.; data curation, S.L.; writing—original draft preparation, S.L. and Y.K.; writing—review and editing, S.L. and Y.K.; visualization, S.L.; supervision, Y.K.; project administration, Y.K.; funding acquisition, B.K. and Y.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly supported by Regional Innovation Strategy (RIS) through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) (Grant No.: 2022RIS-005) and as a project for Cooperative R&D between Industry, Academy, and Research Institute (S3104657) funded by the Ministry of SMEs and Startups (MSS, Korea).

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki and approved by the Institutional Review Board of the Yonsei University Mirae Campus on 17 September 2022 (1041849-202109-BM-139-02) on 17 October 2022.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author. The data are not publicly available because the authors are continuing the study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Moreau, H.; Hélie, S.; Moreau, H.; Arsenault, A. Dockless e-Scooter: A Green Solution for Mobility? Comparative Case Study between Dockless e-Scooters, Displaced Transport, and Personal e-Scooters. *Sustainability* **2020**, *12*, 1803. [CrossRef]
2. Ishaq, M.; Ishaq, H.; Nawaz, A. Life Cycle Assessment of Electric Scooters for Mobility Services: A Green Mobility Solution. *Int. J. Energy Res.* **2022**, *46*, 20339–20356. [CrossRef]
3. Møller, T.; Simlett, J.; Mugnier, E. *Micromobility: Moving Cities into a Sustainable Future*; EY: London, UK, 2020.
4. Nimble App Genie. E-Scooter Statistics: Key Facts and Figures for 2024. Nimble App Genie. 2024. Available online: https://www.nimbleappgenie.com/blogs/escooter-statistics/ (accessed on 2 August 2024).
5. Grand View Research. Scooter Market Size, Share & Trends Analysis Report by Product Type (Normal Scooter, Electric Scooter), by Electric Scooter Type (Conventional Electric Scooter, Swappable Electric Scooter), by Region, and Segment Forecasts, 2023–2030. Grand View Research. 2023. Available online: https://www.grandviewresearch.com/industry-analysis/scooters-market (accessed on 2 August 2024).
6. Grand View Research. Electric Scooters Market Size, Share & Trends Analysis Report by Battery (Lithium-Ion, Lead-Acid), by Drive Type (Belt Drive, Hub Motor), by End-Use (Personal, Commercial), by Region, and Segment Forecasts, 2024–2030. Grand View Research. 2023. Available online: https://www.grandviewresearch.com/industry-analysis/electric-scooters-market (accessed on 2 August 2024).
7. Korea Road Traffic Authority. Traffic Accident Analysis Report 2023. Korea Road Traffic Authority. 2023. Available online: https://www.koroad.or.kr/main/board/6/87946/board_view.do?&cp=1&listType=list&bdOpenYn=Y&bdNoticeYn=N (accessed on 2 August 2024).
8. Namiri, N.K.; Lui, H.; Tangney, T.; Allen, I.E.; Cohen, A.J.; Breyer, B.N. Electric Scooter Injuries and Hospital Admissions in the United States, 2014–2018. *JAMA Surg.* **2020**, *155*, 357. [CrossRef] [PubMed]
9. Posirisuk, P.; Baker, C.; Ghajari, M. Computational prediction of head-ground impact kinematics in e-scooter falls. *Accid. Anal. Prev.* **2022**, *167*, 106567. [CrossRef] [PubMed]
10. Ahn, S.; Kim, J.; Koo, B.; Kim, Y. Evaluation of inertial sensor-based pre-impact fall detection algorithms using public dataset. *Sensors* **2019**, *19*, 774. [CrossRef] [PubMed]
11. Sucerquia, A.; López, J.D.; Vargas-Bonilla, J.F. SisFall: A Fall and Movement Dataset. *Sensors* **2017**, *17*, 198. [CrossRef] [PubMed]
12. Kim, D.; Lee, S.; Koo, B.; Yang, S.; Kim, Y. Threshold-Based Pre-Impact Fall Detection and Its Validation Using the Real-World Elderly Dataset. *J. Biomed. Eng. Res.* **2023**, *44*, 384–391. [CrossRef]
13. Yu, X.; Jang, J.; Xiong, S.; Yu, X. A Large-Scale Open Motion Dataset (KFall) and Benchmark Algorithms for Detecting Pre-Impact Fall of the Elderly Using Wearable Inertial Sensors. *Front. Aging Neurosci.* **2021**, *13*, 692865. [CrossRef] [PubMed]
14. Klenk, J.; Schwickert, L.; Palmerini, L.; Mellone, S.; Bourke, A.; Ihlen, E.A.; Kerse, N.; Hauer, K.; Pijnappels, M.; Synofzik, M.; et al. The FARSEEING real-world fall repository: A large-scale collaborative database to collect and share sensor signals from real-world falls. *Eur. Rev. Aging Phys. Act.* **2016**, *13*, 8. [CrossRef] [PubMed]
15. Kim, Y.; Jung, H.; Koo, B.; Kim, J.; Kim, T.; Nam, Y. Detection of pre-impact falls from heights using an inertial measurement unit sensor. *Sensors* **2020**, *20*, 5388. [CrossRef] [PubMed]
16. Boubezoul, A.; Espié, S.; Larnaudie, B.; Bouaziz, S. A simple fall detection algorithm for powered two wheelers. *Control Eng. Pract.* **2013**, *21*, 286–297. [CrossRef]
17. Yu, X.; Koo, B.; Jang, J.; Kim, Y.; Xiong, S. A Comprehensive Comparison of Accuracy and Practicality of Different Types of Algorithms for Pre-Impact Fall Detection Using Both Young and Old Adults. *Meas. J. Int. Meas. Confed.* **2022**, *201*, 111785. [CrossRef]
18. Koo, B.; Yu, X.; Lee, S.; Yang, S.; Kim, D.; Xiong, S.; Kim, Y. TinyFallNet: A Lightweight Pre-Impact Fall Detection Model. *Sensors* **2023**, *23*, 8459. [CrossRef] [PubMed]
19. Sanchez-Iborra, R.; Bernal-Escobedo, L.; Santa, J.; Skarmeta, A. TinyML-Based Fall Detection for Connected Personal Mobility Vehicles. *Comput. Mater. Contin.* **2022**, *71*, 3869. [CrossRef]
20. Yu, X.; Wan, J.; An, G.; Yin, X.; Xiong, S. A Novel Semi-Supervised Model for Pre-Impact Fall Detection with Limited Fall Data. *Eng. Appl. Artif. Intell.* **2024**, *132*, 108469. [CrossRef]
21. Iguchi, Y.; Lee, J.H.; Okamoto, S. Enhancement of Fall Detection Algorithm Using Convolutional Autoencoder and Personalized Threshold. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021. [CrossRef]
22. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531. [CrossRef]
23. An, J.; Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Spec. Lect. IE* **2015**, *2*, 1–18.

24. Tamura, T.; Yoshimura, T.; Sekine, M.; Uchida, M.; Tanaka, O. A wearable airbag to prevent fall injuries. *IEEE Trans. Inf. Technol. Biomed.* **2009**, *13*, 910–914. [CrossRef] [PubMed]
25. Attal, F.; Boubezoul, A.; Oukhellou, L.; Cheifetz, N.; Espie, S. The Powered Two Wheelers fall detection using Multivariate CUmulative SUM (MCUSUM) control charts. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014.