Author Author

Pre-Impact_Fall_Detection

Turnitin

Document Details

Submission ID	trn:oid:::3618:120464998
Submission Date	Nov 9, 2025, 2:48 PM GMT+5
Download Date	Nov 9, 2025, 2:50 PM GMT+5
File Name	Pre-Impact_Fall_Detection.docx
File Size	34.9 KB

13 Pages

3,228 Words

19,901 Characters

2% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text

Match Groups

6 Not Cited or Quoted 2%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

1%	Internet sources
1%	Publications
0%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **6** Not Cited or Quoted 2%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1%	Internet sources
1%	Publications
0%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	**encyclopedia.pub**	<1%
2	Internet	**smart-app.readthedocs.io**	<1%
3	Publication	**Ton Duc Thang University**	<1%
4	Internet	**vtechworks.lib.vt.edu**	<1%
5	Publication	**Nguyen-Ngan-Ha Lam, Chiao-Hsin Lin, Yi-Lu Li, Wei-Siang Ciou, Yi-Chun Du. "An Io...**	<1%
6	Internet	**www.next.gr**	<1%

A Deep Learning Framework for Pre-Impact Fall Detection on a Wearable IoT Platform

Abstract

Falls among the elderly are a critical public health issue, with significant risk tied to the "long-lie" period following an impact.[1] Current fall detection systems (FDS) are largely reactive (post-impact), while proactive (pre-impact) systems, which offer a "lead time" for intervention, often suffer from high false-positive rates, limiting their reliability.[2] This paper proposes the complete analytical and architectural framework for a novel, wearable IoT system designed for high-accuracy, pre-impact fall detection. The proposed system solves the core trade-off between latency and specificity by implementing a deep learning (DL) model directly on a low-power microcontroller (an "Edge AI" or "TinyML" architecture).[4] This on-device inference eliminates network latency, enabling detection within the sub-400ms pre-impact window.[5] This framework analyzes the required hardware (a 6-axis IMU [8] and ESP32-S3 [9]), data pipeline methodologies (sampling rate analysis [10], windowing strategies [11]), and a comparative analysis of DL models. It concludes that a Transformer-based architecture [12] is the optimal choice, not only for its high accuracy but for its computational efficiency and superior real-world deployment performance compared to traditional RNNs.[12] The report also details the "TinyML" deployment workflow (using TensorFlow Lite [14]) and identifies the system's core patentable novelty as the unique combination of an on-device Transformer model and advanced data generation (e.g., GANs [15]) to achieve a low-latency, high-specificity pre-impact FDS.

Keywords

Fall Detection, Pre-Impact Detection, Deep Learning, Internet of Things (IoT), Wearable Sensors, TinyML, Edge AI, Accelerometer, Gyroscope, Transformer Model, TensorFlow Lite, SisFall Dataset

I. Introduction

The global population is aging at an unprecedented rate; by 2050, the number of individuals aged 60 and over is expected to nearly double.[16] This demographic shift presents a major public health challenge, as falls are a leading cause of unintentional injury and mortality in this population.[16] Falls can lead to severe injuries and a reduction in physical and sensory functionalities.[17] Beyond the initial trauma, a significant portion of the medical risk is associated with the "long-lie"—the period a person remains on the ground, unable to summon help.[1] This delay in assistance exacerbates health complications, making the development of robust, automatic monitoring systems a critical priority.[16]

The primary objective of an automatic fall detection system (FDS) is to minimize the latency between the fall event and the notification of a caregiver.[1] Current systems, however, are almost entirely *reactive*, identifying a fall only *after* the impact has occurred.[17] This approach, while solving the "long-lie" problem, fails to prevent the injury itself.

The state-of-the-art in this field is **pre-impact fall detection**: a proactive system designed to detect a fall *during* the irreversible, uncontrolled descent phase but *before* the user impacts the ground.[3] This capability creates a "lead time"—a window of approximately 300-400 ms [5]—that enables proactive interventions, such as the deployment of wearable airbags.[3]

4 The objective of this paper is to research and define a complete framework for a high-efficacy, wearable IoT system for *pre-impact fall detection*. This system will leverage 6-axis inertial sensors and advanced deep learning models deployed directly on an edge-computing device. This document constitutes the foundational research, providing a comprehensive literature review, a detailed system architecture and data methodology, a comparative analysis of state-of-the-art deep learning models, a concrete deployment plan, and a thorough investigation of the system's patentability.

II. Literature Review and Related Work

A. Taxonomy of Fall Detection Systems

FDS research is broadly categorized into fall prevention and fall detection.[17] Fall prevention systems are prognostic, using long-term monitoring to identify high-risk individuals.[17] Fall detection systems are event-driven and can be:

1. **Reactive (Post-Impact):** These systems, the most common type, identify a fall *after* impact, aiming to reduce the "long-lie" time.[17]
2. **Proactive (Pre-Impact):** This emerging class aims to detect the fall *before* impact.[3] Early pre-impact systems relied on simple thresholding, which could be fast but suffered from high false alarms, making them unreliable for practical use.[2]

B. Sensor Modalities for Fall Detection

While ambient sensors (e.g., cameras) exist, wearable sensors are the mainstream solution due to their mobility.[2]

- **Accelerometer-Only Systems:** Many early systems used only a 3-axis accelerometer. However, these are notoriously susceptible to false positives from fall-like Activities of Daily Living (ADLs) such as sitting down quickly.[2]
- **6-Axis IMU (Accelerometer + Gyroscope):** A fall is a complex kinematic event defined by *both* acceleration and *rotational body tilt*.[8] A gyroscope is required to measure this angular velocity. Research confirms that fusing data from both accelerometers and gyroscopes dramatically enhances detection accuracy and robustness.[8]

C. System Architectures: Cloud vs. Edge

The location of data processing is a critical design choice.

- **Cloud-Based Architecture:** In this model, the wearable streams raw sensor data to a cloud server for processing. This architecture is non-viable for pre-impact detection. The round-trip network latency, combined with server-side computation, will certainly exceed the 300-400 ms pre-impact lead time.[4]
- **Edge-Based (TinyML) Architecture:** This model performs all deep learning inference *directly on the wearable device's microcontroller*.[4] This "TinyML" approach solves the latency problem, as inference is performed in microseconds. It also provides superior privacy, reliability, and low power consumption.[18]

D. Deep Learning Models for Human Activity Recognition (HAR)

Deep learning has replaced manual feature engineering in HAR.[17] Common models include 1 1D-Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU).

III. Proposed System Architecture and Methodology

This section details the proposed hardware and software architecture designed to meet the goal of low-latency, high-accuracy pre-impact detection.

A. Sensor Selection

The system requires a 6-axis IMU. The proposed sensor is the **MPU-6050**, which integrates a 3-axis accelerometer and a 3-axis gyroscope, providing the 6-dimensional data necessary for the deep learning model to distinguish falls from ADLs with high specificity.[8]

B. Processing Unit and Edge Deployment

The processing unit must be power-efficient and capable of running a DL model on-device. The proposed processor is the **ESP32-S3 microcontroller**. This variant is explicitly designed for on-device AI and offers superior power efficiency for machine learning tasks.[9]

C. System Architecture

The system is built on an **Edge-AI (TinyML) architecture**.[4] The deep learning inference occurs *directly* on the ESP32-S3. This architecture (visualized in Figure 1) is the only way to meet the sub-400ms latency requirement for pre-impact detection.

Figure 1: Proposed System Architecture

(Conceptual Diagram: \ --- (Bluetooth Low Energy) ---> \ --- (Wi-Fi/Cellular) ---> \ --->)

D. Communication Protocol

In this edge-based design, communication is only for alert dissemination.

1. **Device-to-Gateway:** The wearable device will use **Bluetooth Low Energy (BLE)** to communicate with a local gateway (e.g., the user's smartphone). BLE is the industry standard for power-efficient, short-range communication in wearables.[19]
2. **Gateway-to-Cloud:** Once the gateway receives the "FALL_DETECTED" signal, it relays this tiny alert packet to the cloud. The **Message Queuing Telemetry Transport (MQTT)** protocol is ideal for this task, offering low-latency, secure dissemination.[20]

IV. Datasets and Data Processing Pipeline

A. Dataset Selection and Analysis

A robust, high-quality, and well-labeled public dataset is required for training. The primary benchmark for this project will be **SisFall.**[10]

SisFall is widely considered the best-in-class dataset because it is the largest, was captured with a high-fidelity 200 Hz 6-axis IMU, includes 38 volunteers (including elderly participants), and features a wide variety of 15 fall types and 19 ADLs.[21] Compared to other public datasets such as MobiFall and UMAFall, SisFall has the largest amount of data.[23] Its use in other pre-impact studies makes it a valid benchmark.[7] Table 1 provides a comparison.

Table 1: Comparative Analysis of Public IMU Fall Datasets

Dataset Name	Primary Citation(s)	Sensor(s)	Sampling Rate	No. of Participants (Age Range)	No. of Fall Types	No. of ADLs	Key Feature/Limitation
SisFall	Sucerquia et al. (2017) [21]	2 Accelerometers, 1 Gyroscope	200 Hz [10]	38 (23 young, 15 elderly) [22]	15	19	**Best-in-class:** High-fidelity, includes elderly, high variety.
MobiFall	Vavoulas et al. (2013) [23]	Accelerometer, Gyroscope, Orientation	~87 - 100 Hz [23]	24 volunteers	4	9	Smartphone-based, less sensor control, fewer fall types.
UMAFall	Casilari et al. (2017) [23]	Multi-sensor (IMU, smartphone)	Variable	17	3	5	Multi-sensor, but smaller scale than SisFall.

B. Data Imbalance and Augmentation

Fall detection is a classic imbalanced learning problem: fall samples are vastly outnumbered by ADL samples. Training on this data naively will result in a model that misses falls.[24] To address this, we will implement advanced augmentation techniques, specifically **Generative Adversarial Networks (GANs)**. A GAN can be trained on the *real* fall data to learn its complex distribution and generate an unlimited supply of new, synthetic, and highly realistic fall-data sequences, outperforming other techniques.[15]

C. Data Preprocessing: Sampling Rate Analysis

The choice of sampling frequency is a critical trade-off.

5

- **Biomechanical Argument:** Analysis of fall events shows the vast majority of the signal's energy is concentrated in a low-frequency band (2-3.5 Hz) [25], suggesting a low sampling rate is sufficient.
- **Kinematic Argument:** Higher sampling rates (e.g., SisFall's 200 Hz [10]) capture finer-grained kinematic details.

While studies show that 50 Hz is generally sufficient for accurate detection [1], the "comparative result analysis" for this project *must* test this. The methodology is to take the 200 Hz SisFall data, downsample it (to 100 Hz, 50 Hz, and 20 Hz), and then train and evaluate the same DL models on all versions to find the optimal balance.[1]

D. Data Preprocessing: Windowing Strategy

DL classifiers do not operate on continuous data. The stream must be segmented into fixed-size "windows."

1. **Technique:** The standard method is the **Fixed-Size Overlapping Sliding Window (FOSW)**.[11] A window of a fixed duration (e.g., 2 seconds) is extracted. The window then "slides" forward by a "step size" (e.g., 1 second), creating an overlap.
2. **Pre-Impact Labeling:** For pre-impact detection, the data must be meticulously re-labeled into three classes: (1) **Non-fall/ADL**, (2) **Pre-impact**, and (3) **Post-impact/Fall**.[24]
3. **Window Size vs. Lead Time Trade-off:** The *size* of the window is a critical variable. A small window (e.g., 200-500 ms) is processed faster, potentially increasing lead time, but contains little context, likely increasing false positives.[24] A large window

provides rich context but may be too slow, eliminating the pre-impact lead time. This trade-off must be experimentally validated.

V. Model Architectures for Comparative Analysis

The core of the project is a "comparative result analysis" of DL models. This analysis must evaluate models not just on their *offline* accuracy but on their *suitability for real-time, on-device (TinyML) deployment*.

A. Baseline and Hybrid Models

The analysis will include baseline models such as 1D-CNNs and RNNs (LSTMs, GRUs) [17], as well as hybrid CNN-LSTM models.

B. Advanced Models (Proposed SOTA)

- **The Transformer Model:** This architecture relies on a **self-attention mechanism**. This allows the model to "pay attention" to the most critical parts of the data window, making it powerful for distinguishing falls from fall-like ADLs. Transformers have achieved SOTA accuracy (99.8%) on the SisFall dataset.[13]

C. Justification for Transformer Deployment

The **Transformer** is the hypothesized optimal choice, not just for its accuracy, but for its *deployment efficiency*.

- **The "Offline vs. Online" Contradiction:** Studies explicitly show that while a CNN-LSTM may achieve *higher*accuracy in *offline* testing, the **Transformer** is the *preferable choice for real-time deployment*.[12]
- **Performance Transfer:** One study noted a CNN-LSTM's F1-score dropped from 87.6% (offline) to 70.2% (real-world), while the Transformer's performance remained stable, indicating it can better transfer its results to a real-world device.[12]
- **Parallelization:** LSTMs/GRUs are sequential and slow. Transformers are highly *parallelizable*, making them more computationally efficient and *faster* at inference time—a massive advantage for a resource-constrained edge device.[12]

This project's hypothesis is that the **Transformer will provide the best *real-world, on-device* balance of accuracy, specificity, and pre-impact lead time**.

VI. Evaluation Metrics and Benchmarks

A. Standard Classification Metrics

For an imbalanced dataset, "accuracy" is a misleading metric.[24] The evaluation must be based on the confusion matrix, focusing on the trade-off between:

- **Sensitivity (Recall):** (True Positives / (TP + FN)). The most important medical metric. It answers: "Of all *actual falls*, what percentage did our system *correctly detect*?".[26]
- **Specificity:** (True Negatives / (TN + FP)). The "user annoyance" metric. It answers: "Of all *normal ADLs*, what percentage did our system *correctly ignore*?".[1]
- **F1-Score:** The harmonic mean of Precision and Recall ($2 \cdot (\text{Precision} \cdot \text{Recall}) / (\text{Precision} + \text{Recall})$). An excellent single-number summary of performance on the minority (fall) class.[26]

B. The Novel Pre-Impact Metric: Lead Time

For this project's *novel* goal, the metrics above are insufficient. The single most important new metric is **Lead Time**.

- **Definition:** Lead Time is the duration, in milliseconds (ms), from the moment the FDS algorithm first triggers a "pre-impact fall" alert to the moment of actual ground impact.[5]
- **Benchmark:** This is the *only* window available for proactive intervention. Literature on threshold-based pre-impact systems reports average lead times of **323 ms** [6] to **427 ms**.[5] The goal of this deep learning project is to *match or exceed* this lead time, while *dramatically increasing* the specificity that plagues simpler methods.[7]

C. Benchmark Tables

The project's final "comparative result analysis" should be presented against established benchmarks from the literature, such as those in Table 2 and Table 3.

Table 2: Comparative Performance Benchmarks on SisFall Dataset (Post-Impact)

Model Architecture	**Citation(s)**	**Sensitivity (%)**	**Specificity (%)**	**F1-Score (%)**
Transformer (Attention)	[13]	99.87	N/A	~99.8

Other models (e.g., CNN, LSTM)	[2]	*Various*	*Various*	*Various*

Table 3: Evaluation of Pre-Impact Detection Performance

Model/Algorithm	**Citation**	**Avg. Lead Time (ms)**	**Specificity (%)**	**Key Trade-Off**
Vertical Velocity Threshold	[6]	323 ms	100% (on their ADLs)	Simple, but high false alarms on unseen ADLs.
Triangle Feature (TF) Algorithm	[5]	427 ms	83.9% (on SisFall)	Best lead time, but still 16% of ADLs cause false alarms.
CNN-LSTM (Project)	(Experimental)	*To Be Measured*	*To Be Measured*	*Hypothesis: Good specificity, but lower lead time?*
Transformer (Project)	(Experimental)	*To Be Measured*	*To Be Measured*	*Hypothesis: Best balance of lead time and specificity?*

VII. On-Device Deployment Workflow (TinyML)

This section provides the *execution plan* for the project. The deployment will use the **TensorFlow Lite (TFLite)**framework, the industry standard for on-device machine learning.[14]

1. **Step 1: Training:** The model (e.g., the Transformer) is designed, trained, and validated in a high-level framework like TensorFlow/Keras.[14]
2. **Step 2: Conversion:** The saved Keras model is fed into the **TensorFlow Lite (TFLite) converter**. This converts the complex model into a highly efficient, portable .tflite FlatBuffer file.[14]
3. **Step 3: Quantization:** This is a critical optimization step. Post-training quantization is applied, converting the model's 32-bit floating-point weights into 8-bit integers. This drastically reduces the model's size and makes it run significantly faster on microcontrollers.
4. **Step 4: On-Device Deployment:** The final, quantized .tflite model file is deployed onto the ESP32. The device's firmware uses the **TensorFlow Lite Micro (TFLM)** library, a minimal C++ library designed to run TFLite models on microcontrollers with extremely small memory.[14]

VIII. Analysis of Novelty and Patentability

A patent is granted for an invention that is *novel*, *non-obvious*, and *useful*. This section identifies the specific, patentable claims of this project's "idea."

A. Survey of Prior Art (What is *Not* Novel)

The individual components are likely "prior art" and not patentable:

- Using an accelerometer for fall detection is well-established.
- Using a 6-axis IMU (accelerometer + gyroscope) is also known.[8]
- Simple threshold-based methods for pre-impact detection exist.[6]
- Using TFLite for on-device deployment is also known.[14]

B. The Inventive Step: A Novel *Combination* for a *Specific* Purpose

The invention is *not* in the individual components, but in their **specific, non-obvious combination to achieve a result that the components in isolation could not.** The prior art fails to provide a system that is *both* pre-impact *and* has the high specificity of a deep learning model. Threshold systems [7] have lead time but poor specificity. Cloud-based DL systems [4] have good specificity but *no* lead time.

The invention is the *system architecture* that solves this: a Transformer model, deployed on-device, to achieve *both* pre-impact lead time *and* high specificity.[12]

C. Draft Patent Claims

The core "idea" for a patent application would be structured as follows:

- **Claim 1 (The System):** "A wearable system for pre-impact fall detection, comprising:
  - a) A 6-axis Inertial Measurement Unit (IMU) [8] configured to generate 6-channel time-series data of a user's motion;
  - b) An on-device microcontroller [9] operatively coupled to the IMU; and
  - c) A **Transformer-based** deep learning model [13] deployed on said microcontroller using a lightweight inference framework [14];
  - d) wherein said model is trained to classify said time-series data as a 'pre-impact fall' by processing said data on-device to minimize latency [18];

  - e) and wherein said system generates an alert *prior to* the user's physical impact with a surface, thereby maximizing a 'lead time' metric.[5]"
- **Claim 2 (The Method):** "A method for real-time, pre-impact fall detection on a wearable device, comprising the steps of:
  - a) Receiving 6-axis IMU data from a user;
  - b) Segmenting said data into overlapping windows [11];
  - c) Feeding said windows, *on-device*, into a **self-attention-based** neural network classifier;
  - d) Analyzing said windows with the classifier's self-attention mechanism to identify a kinematic signature of an irreversible, pre-impact fall; and
  - e) Triggering an alert if said signature is detected, wherein said alert is triggered with a lead time of over 300 ms before ground impact.[5]"
- **Claim 3 (The Training Novelty):** "The method of Claim 2, wherein the self-attention-based classifier is trained on a dataset comprising both real fall data and synthetic fall data, wherein said synthetic fall data is generated by a **Generative Adversarial Network (GAN)**.[15]"

IX. Conclusion and Future Work

A. Conclusion

This paper has presented a comprehensive research framework for a novel, pre-impact fall detection system. The analysis concludes that the primary technical and clinical novelty lies in achieving **pre-impact detection** [3] with the high specificity of a deep learning model. This is only possible via an **Edge AI (TinyML) architecture** that performs all inference on-device, as the sub-400ms latency budget [5] makes cloud-based processing non-viable.[4]

The comparative analysis of models concludes that the **Transformer** is the most promising model for this application.[13]This is due not only to its SOTA accuracy but, more critically, to its superior *deployment efficiency* and *real-world performance transferability* when compared to sequential RNN models.[12] Finally, the project's success must be evaluated not on accuracy alone, but on the primary metric of **Lead Time (in ms)**, as this is the metric that quantifies the system's ability to enable proactive intervention.[7]

B. Future Work

The execution path for this project will proceed as follows:

1. **Hardware Prototyping:** Procure and integrate the MPU-6050 and ESP32-S3.[9]
2. **Data Pipeline Implementation:** Implement the full data-processing pipeline. This includes downloading the SisFall dataset [21], re-labeling it into three classes ("ADL," "Pre-impact," "Fall") [24], implementing the FOSW function [11], and creating the downsampled (200Hz, 100Hz, 50Hz) dataset versions.[1]
3. **Advanced Methodology:** Implement a GAN [15] in TensorFlow to generate synthetic "Pre-impact" class data to balance the dataset.
4. **Modeling and Analysis:** Train and benchmark the three models (1D-CNN, CNN-LSTM, Transformer) on the augmented datasets. The results will be compiled into the final "Comparative Analysis" (Tables 2 and 3).
5. **Deployment and Validation:** Select the best-performing model (hypothesized to be the Transformer [12]), use the TFLite converter to quantize and convert it [14], and deploy the final model to the ESP32 to demonstrate real-time, on-device detection.

References

1 https://arxiv.org/html/2505.04660v1

2 https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2020.00071/full

3 https://pmc.ncbi.nlm.nih.gov/articles/PMC8347190/

4 https://pmc.ncbi.nlm.nih.gov/articles/PMC7866865/

5 https://jkkniu.edu.bd/uploads/publication/2024-05-16-3d6dd7_694272408.pdf

6 https://scispace.com/pdf/fall-detection-system-with-artificial-intelligence-based-1qo3phdr.pdf

7 https://pmc.ncbi.nlm.nih.gov/articles/PMC11019185/

8 https://www.mdpi.com/2078-2489/15/3/161

9 https://www.mdpi.com/2227-7080/12/9/166

10 https://pmc.ncbi.nlm.nih.gov/articles/PMC12334619/

11 https://www.mdpi.com/2624-6120/2/1/1

12 https://pmc.ncbi.nlm.nih.gov/articles/PMC11333863/

13 https://pmc.ncbi.nlm.nih.gov/articles/PMC8309569/

14 https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2020.00063/full

15 https://pure.ulster.ac.uk/files/133345559/IEEE_paper_.pdf

16 https://www.researchgate.net/figure/A-cloud-network-edge-architecture-for-fall-detection-prevention-and-protection-1_fig1_335620708

17 https://pubmed.ncbi.nlm.nih.gov/28117691/

18 https://pmc.ncbi.nlm.nih.gov/articles/PMC5298771/

19 https://www.mdpi.com/1424-8220/23/10/4774

20 https://www.mdpi.com/1424-8220/24/19/6235

21 https://www.mdpi.com/1424-8220/19/4/774