A Smart TinyML Framework for Real-Time Earth Tremor Detection in Low-Power IoT Sensor Networks

Manju Bhargavi D P[1], Arpitha K M[2], Ashwini S[3], Bhargavi S[4]
1st Assistant Professor, CSE, RLJIT, Bengaluru, India mbhargavi109@gmail.com
2nd Assistant Professor,CSE(AI-ML),CITNC,Bengaluru,India arpikm1998@gmail.com
3rd M.Tech Scholar, CSE, RLJIT, Bengaluru, India ashigowda.s@gmail.com
4th M.Tech Scholar, CSE, RLJIT, Bengaluru, India bhargavissathish@gmail.com
Corresponding Author: Manju Bhargavi D P*
mbhargavi109@gmail.com

Abstract
Rapid and reliable earthquake detection is essential for minimizing disaster impacts and enabling timely emergency response. Conventional earthquake monitoring systems primarily rely on centralized seismic stations and cloud-based processing infrastructures, which often introduce communication delays, require significant computational resources, and are difficult to deploy in remote or resource-constrained environments. To address these limitations, this study presents a Tiny Machine Learning (TinyML)-enabled edge intelligence framework for real-time earthquake tremor detection on low-power Internet of Things (IoT) devices. The proposed system integrates an ESP32 microcontroller with an MPU6050 accelerometer to perform on-device seismic signal analysis without dependence on cloud connectivity. Seismic waveform data obtained from the Mendeley Mini SEED dataset were pre-processed using down-sampling, normalization, and sliding-window segmentation to generate fixed-length time-series samples suitable for embedded inference. A lightweight one-dimensional Convolutional Neural Network (1D-CNN) was trained to distinguish earthquake events from non-earthquake signals and subsequently optimized using TensorFlow Lite Micro for deployment on the embedded platform. The optimized model was converted into C-compatible code, enabling efficient execution within the limited memory and computational resources of the ESP32 microcontroller. Experimental evaluation demonstrated an overall classification accuracy of approximately **83%**, with competitive precision, recall, and F1-score, confirming the effectiveness of the proposed embedded inference framework. Furthermore, the system achieved low-latency, energy-efficient operation while providing confidence-based real-time alerts through an LCD interface. The proposed TinyML framework demonstrates the feasibility of executing deep learning models directly on microcontroller-class devices, offering a scalable, low-cost, and energy-efficient solution for distributed seismic monitoring, disaster early warning, and smart city applications.

Keyword(s): TinyML, Earthquake Detection, Edge AI, IoT, ESP32, Convolution Neural Networks, Seismic Signal Processing
How to cite this article: Manju Bhargavi DP, Arpitha KM, Ashwini S, Bhargavi S. A Smart TinyML Framework for Real-Time Earth Tremor Detection in Low-Power IoT Sensor Networks. Int J Drug Deliv Technol. 2026;16(68s):1016-1025. DOI: 10.25258/ijddt.16.68s.118

1. Introduction

Earthquakes are considered as some of the most devastating natural risks, which have the potential of producing mass loss of life, extensive destruction of infrastructure and the socioeconomic worst side effects in both developed and developing countries. This characteristic of seismic activities as being unpredictable, alongside the fast release of energy and the massive movement of the ground, demands the use of real-time monitoring and early detection systems as very essential aspects of contemporary disaster management measures. The capability of detecting the seismic activity at the latest phase possible may greatly enhance the coordination of the emergency response, create the possibility of automated safety measures, and ensure the minimal overall effect on the endangered population and the destruction of the critical infrastructure. Conventional mode of earthquake detection and monitoring makes use of a series of seismic sensors that are distributed largely but continuously transmit waveform information to centralized process centres. They are centralized systems, typically backed by high-performance computing systems and cloud-based services, that use signal processing and statistical or machine learning to identify seismic events and determine parameters of such events such as magnitude, depth, and epicentre location. Although these architectures have proved to be very accurate and reliable on a national and global scale, they have intrinsic drawbacks associated with latency, communication overhead, and network availability susceptibility[1]. The usefulness of the centralized monitoring and the distribution of alerts can be significantly lowered in areas with dis-connectivity or even in the case of large-scale disasters, when the network can be damaged.

The current development of Edge Artificial Intelligence (Edge AI) has brought a paradigm shift in the approach to the design and deployment of intelligent systems. Rather than transferring data to a cloud-based method of computation, Edge AI allows executing data processing and decision-

making at data source, potentially a sensor or an embedded device. This decentralized scheme leads to less frequent transmission of data, a reduced response time, and the system robustness will be increased in conditions where the connection is not always accessible and is not always reliable. Edge-based systems are also characterized by better privacy through processing local data and reduced bandwidth usage and decreased cost of operation due to storage and transmission of large volume of data. A closely related and fast-growing area is Tiny Machine Learning (TinyML) which deals with the deployment of machine learning models on ultra-low-power microcontrollers and constrained resources embedded systems. They are normally small-memory, low-computation, and power-constrained, but can be used to infer tasks in real-time and this includes speech recognition, gesture recognition, environmental sensors, and predictive maintenance. The development of lightweight neural network implementations and on-board inference systems like TensorFlow Lite Micro have made the implementation of smart sensing systems such that they can work autonomously over extended durations a reality.[2]

Although the use of Edge AI and TinyML in domains is increasingly growing, their use in classification of seismic signals and earthquake detection have yet to reach their full potential, especially when deployed with fully embedded real time hardware. The majority of current studies in machine learning-based seismic analysis belong either to the field of offline seismic processing or to cloud-based inference, in which computational and memory limitations are not as limiting. Although such strategies are effective in terms of classification accuracy, they do not necessarily have to be implemented in distributed sensing systems in remote, low-powered, and deployed locations of large scales.

The current study fills this gap by suggesting an independent, real-time earthquake detection device, which would carry out an on-the-machine machine learning inference on a low-power Internet of Things (IoT) platform. It features an ESP32 microcontroller, a low-priced inertial sensor, and a lightweight one-dimensional Convolutional Neural Network (1D-CNN) model as a preferred scheme in deploying the system. The offered architecture eliminates the reliance on external computing infrastructure and, therefore, allows local, autonomous seismic activity detection, which is especially appropriate when the architecture is implemented in intelligent buildings, campuses, and one-to-many sensor networks. The suggested system is also streaming-based signal processing structure, when the continuous vibration data is divided into time windows with a fixed length and processed in real time. This strategy is a reflection of the feasibility of embedded hardware and enables the trained model to be tested in the real-world deployment settings. Combination of sliding window segmentation method, normalization and down sampling technique will ensure that the input representation is aligned with the training data as well as the minimal computational overhead.

The overall intention behind this study is to come up with scalable and cost-effective seismic observations that can be used to supplement the current professional seismic networks. Although the suggested system is not supposed to substitute high-precision seismological equipment, it shows the realness of implementing smart sensors nodes that can work both independently and integrated into a distributed sensorial system. Such systems have the potential to improve the coverage in areas with low capacity to modern Smart Cities infrastructures, as well as to enable localised early-warning mechanism in critical facilities (schools, hospitals and industry sites).

Some of the main contributions of the work are the following ones:

1. **TinyML pipelines design at the extreme end: Seismic signal classification:**

These translate to full workflow, acquisition of data, data recovery, data training, data optimization and embedded deployment.[3]

2. **Real-time Signal Processing Preprocessing and Windowing Framework:**

It deploys a streaming-based architecture that allows forwarding vibration signals on microcontroller platform in real-time and permit continuous segmentation and normalization of vibration signals.[4]

3. **Deployment of a Lightweight CNN Model Using TensorFlow Lite Micro:**

It is a trained model optimized and work integrated into the ESP32 firmware to infer on-device with a firm resource allocation of a rigid memory and power budget.[3]

4. **System Evaluation on Application of Embedded Hardware:**

The evaluation of performance is considered based on standard machine learning measurements as well as real-time check-ups using hardware, such as latency, memory consumption, and alert stability.[5]

2. Literature Review

The conventional earthquake sensing systems utilize systems of high-resolution seismometers that have centralized signal processing units. The high-performance computing systems such as the United States Geological Survey (USGS) and the Shake Alert utilize the data collected on the waveforms of the shaking and issue warnings. The recent interest in seismic signal classification based on machine learning methods is numerous. Nonetheless, little studies have been conducted concerning the

implementation of TinyML in real-time hardware seismic detection.

1. Yoon, C. E., O'Reilly, O., Bergen, K. J., and Beroza, G. C. (2015)., Earthquake Detection through Machine Learning.

Proceedings of the National Academy of Sciences (PNAS). This experiment lies at the foundation of one of the first massive uses of machine learning to detecting seismic events. The authors illustrate the ability of the commercial models of supervised learning to differentiate between the earthquake signals and background noise based on the features of the waveforms.

2. Ross, Z. E., Meier, M. A., & Hauksson, E. (2018)., P Wave Arrival Picking and First-Motion Polarity Determination by Deep Learning. Journal of Geophysical Research: Solid Earth.

In this paper, a deep learning architecture is presented to identify P-wave arrivals and determine the characteristics of seismic events in an accurate manner. This study illustrates the excellence of the convolutional neural networks, compared to the standard signals processing techniques, highlighting the usefulness of the deep learning in uncovering temporal motifs on the earthquake time-series records.

3. Perol, T., Gharbi, M., & Denolle, M. (2018)., Convolutional Neural Network for Earthquake Detection and Location., Science Advances.

The authors suggest a CNN-based consummation that can recognize and localize earthquakes directly out of the waveform records. It demonstrates that end to end deep learning models can be used to handle the relocation of the hand-crafted feature extraction and this gives a great reason to consider the application of CNNs in the real time seismic monitoring systems.

4. Mousavi, S. M., Sheng, Y., Zhu, W., and Beroza, G. C. (2019)., A Deep Residual Network of Convolutional and Recurrent Units for Earthquake Signal Detection., Scientific Reports

The paper introduces an example of a hybrid deep learning model that can be used to accurately detect earthquakes by using CNN and recurrent layers. The model has high generalization to other seismic stations further supporting the significance of deep temporal feature learning when performing seismic signal classification.

5. Banbury, P., Reddi, V. J., Torelli, P., and others (2020)., Micronets: Neural Network Architectures for Deploying TinyML Applications., The IEEE International Conference on Computer Design (ICCD), Beijing, China on 7 October 2011.

This paper will discuss neural network architectures that have been optimized to run on microcontrollers. It gives performance and memory numbers on tiny ML systems and gives fundamental information on model design trade-offs in the context of deploying CNNs to such systems as ESP32.

6. Warden, P. & Situnayake, D. (2019)., TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-low Power Microcontrollers., O'Reilly Media (Book / Reference Work)

This reference work provides details on practical deployment of machine learning models on microcontroller platform where TensorFlow Lite Micro is implemented. It offers implementation strategies, optimization methods, and memory management techniques and which are directly informed in the embedded deployment approach, adopted in this study.

7. Allen, R. M. & Melgar, D. (2019)., Earthquake Early Warning: Advances, Scientific Challenges, and Societal Needs., Annual Review of Earth and Planetary Sciences

The paper discusses the recent advancement of the earthquake early warning systems, sensor networks, data processing pipeline, and communication issues. It offers a background on why low-latency, decentralized detection systems like the edge-based system offer in this study are important.

8. Lane, N. D., Bhattacharya, S., Mathur, A., and others (2017)., Squeezing Deep Learning into Mobile and Embedded Devices., IEEE Pervasive Computing.

The authors write about how to perform the deployment of the deep learning models to the hardware that is resource-constrained, such as model compression, quantization, and architectural optimization. The work has a theoretical and practical foundation of TinyML models implementation on microcontrollers.

9. Magana-Zook, S. A., Clayton, R. W. and Cochran, E. S. (2020)., Seismic Event Classification in machine learning for Low Power Sensor Networks, Sensors (MDPI)

This study explores the applications of machine learning work in the classification of seismic events in distributed and low powered sensor networks. These findings confirm the practicability of decentralized detection systems and are in line with the objectives of the system level design of the proposed architecture based on TinyML.

10. Latif, M., et al. (2021)., Edge AI for Smart City Applications: A Survey., IEEE Access

The current survey paper revises the importance of edge-based artificial intelligence in a smart city setting such as environmental monitoring and disaster management. It gives a bigger view of the system level that makes the proposed earthquake detection system very relevant with the new smart infrastructure ecosystems.

3. System Architecture

3.1 Overview

The given system is organized in terms of a layered system, which incorporates seismic data acquisition, intelligent signal classification, and integrated embedded deployment into one real-time monitoring system.

The first layer, which is known as the **Data Layer,** has the role of recording and delivering seismic waveform inputs whether in pre-recorded seismic datasets, or in signal vibrations obtained in real-time by onboard sensors. This layer provides a coherent stream of time-series data which is a reflection in a real physical seismic and environmental state of the world. The second one is the **Intelligence Layer** that applies convolutional neural network (CNN)-based classification model that is used to identify earthquake signals and background noise. The layer: It is used to extract temporal features and conduct probabilistic classifications using a fixed length signal window so that the seismic activity can be detected robustly and flexibly. Intelligence layer is created and tested in offline system through high-level machine learning systems and then optimized to fit into embedded system. The third layer, which is also known as the **Edge Deployment Layer,** has the role of implementing the trained and optimized model directly into an ESP32 microcontroller. This layer will combine inference engine with the peripheral devices, such as display, and audio alert system, to offer real time feedback. The system removes the need to use cloud-based infrastructure and performs inference at the edge, allowing localized and low-latency detection applicable to deployed in distributed and resource-constrained systems.[3]

3.2 Hardware Components

The main hardware unit of the system is the ESP32 microcontroller, which serves as the central processing unit of both sensor data collection and real-time model, and inferences. The ESP32 has adequate processing units, system memory and inbuilt communication devices to facilitate continual sampling, sliding window processing, and embedded neural network implementation.[5] The MPU6050 accelerometer, which is a low-cost inertial measurement device, is used in seismic vibration sensor. This sensor gives high-resolution motion data that is computed to obtain the magnitude of acceleration vector which is used in machine learning classification as a one-dimensional time-series signal.[5] The system uses a 16x2 lcd display tied via the I2C interface to display user visible feedback (confidence scores and status of the detection). This graphical user interface increases the interpretability and monitoring of this system in the validation and deployment of the system under experimentation.[5] The 5 V USB powering is regulated to provide constant and always stable power to the peripheral components and the microcontroller. Such power setup provides a stable long-term operation and can be extended to battery-powered or solar-assisted deployment in case of the future.[5]

3.3 Software Stack

Software of the system is split to offline formulation of models and firmware run of the embedded systems. The entity that processes the data required to feed through the seismic waveform files provided in miniSEED format is Python along with the ObsPy library that handles the processing of the datasets, signal preprocessing, and the formation of features based on windows of the signal preparation. This is an environment that offers the means for efficient processing of large-scale time series data and reproducible processing pipelines.[6]

The convolutional neural network architecture is designed and evaluated with the help of Machine learning model that is developed and trained with the help of TensorFlow and Keras, which offer an adaptable framework to use. These are assisting in fast experimenting, texting hyperparameters, and appraising execution with standard evaluation of classification metrics.[2] To be able to work with embedded deployment, the trained model can be converted to a lightweight multiplexing, and the neural network can be directly executed on the ESP32 microcontroller without the need to allocate memory dynamically or use operating systems. This ensures that all perform determinism and utmost efficiency of the memory utilization process.

Finally, it involves the Arduino Framework in the development of the firmware, the abstraction of hardware and assembly of the sensor, display and the inference engine.

4. Description of Dataset and Preprocessing of Data.

The data utilized in the present paper was acquired at the Mendeley Data Repository and included seismic waveforms in miniSEED format, a common standard in seismological studies to encode sensor data of time-series. The data consists of two major classes: earthquake signals, which are the real seismic activity, and background noise signals, which are non-seismic vibrations of the environment as well as the instrument. Each waveform is registered as a high frequency time series recorded by seismic sensors, which reflects non-stop ground

motion over very long periods of time. In order to facilitate the machine learning model to classify and be compatible with the embedded deployment resources, the raw seismic data was processed through a sequence of structured preprocessing that was used to produce homogeneous, fixed-length input samples. This process takes the continuous stream of waveforms and converts it to pre-defined time windows for convolutional neural network (CNN) training and real-time inferencing.[2]

Signal down sampling was necessary as the first level of preprocessing. The sampled original data were down sampled with an anti-aliasing decimation filter to 50 Hz. The step captures the dominance of frequency content of seismic vibrations and dramatically minimizes the complexity of computations and memory needs.

Amplitude normalization was performed in each segmented window after the down sampling was done. The signal values had been adjusted to a uniform range of between -1 to 1, and this lessened the effects of sensor relocation, surroundings and signal strength. This process of normalisation provides greater stability of a model.[4]

Sliding window segmentation strategy was applied in the last stage of preprocessing. Continuous seismic signals were divided into non-overlapping, fixed length, 4-second (200 samples) windows of preceding and following windows were 50% overlapped. This method is known to amplify the apparent size of the existing training data and still retain the temporal continuity and highly comparable to the real-time streaming feature of the integrated system.[3]

5. Model Architecture

Temporal patterns were extracted from seismic waveforms using a light-weight 1D Convolutional Neural Network (CNN) which was designed for the purpose by

Table 5.1: Model Architecture Synopsis.

Layer	Configuration	Output Shape
Input	200 × 1 time-series window	200 × 1
Conv1D + ReLU	16 filters, kernel = 3	200 × 16
Max Pooling	Pool size = 2	100 × 16
Dropout	Rate = 0.2	100 × 16
Conv1D + ReLU	32 filters, kernel = 3	100 × 32
Max Pooling	Pool size = 2	50 × 32
Flatten	—	1600
Dense + ReLU	16 neurons	16
Output (Sigmoid)	1 neuron	1

A Convolutional Neural Network (CNN) (1D, lightweight) was adapted to simulate the temporal features of seismic waveforms with the aim of being offered on high-energy IoT microcontrollers as a part of TinyML. The network takes a time-series input window of 200 x 1 size and uses two convolutional blocks with 16 and 32 filters respectively, each respectively succeeded by ReLU activation, and max-pooling to successively compute hierarchical features of temporal variations and dimensional-reduction. Drop out layer (rate = 0.25) is taken to achieve better generalization and prevent overfitting of the network. The feature maps that are obtained are flattened and fed through a small dense layer of 16 neurons to learn discriminative features. Lastly, a sigmoid actuated output neuron carries out binary classification to differentiate tremor occurrence and the background noise. The architecture had custom-made depth and parameter efficiency to compromise between detection performance and memory footprint and real-time ability to infer suitable for the resource-constrained edge device. This architecture trades off performance in classification with memory utilization to deploy microcontrollers.

6. Methodology

In this section, a systematic plan of how the proposed TinyML-based real-time earthquake detection system was to be designed, implemented, and evaluated is described. The methodology has combined seismic signal processing, machine learning model development, and embedded system deployment to be one end-to-end framework.

6.1 System Design Approach

The suggested methodology is further subdivided into three large stages:

1. **Offline Model Development Stage**

This step involves the acquisition of data, preprocessing of the signals, creation of windows, and model training based on the high-level machine learning frameworks.

2. **Model Optimization and Conversion Phase**

During this step, the trained neural network is translated into a low-weight and embedded-friendly format via TensorFlow Lite and made optimized in terms of memory and computational resources.

3. **On-Device Deployment /Real-Time Inference Phase**

It will consist of implementing the improved model into the ESP32 code, obtaining real-time sensor readings, conducting real-time inferences, and creating alerts to be shown to users. This layered design provides for modularity, reproducibility (interoperability) and scalability for future extensions of the design like cloud integration or multi-node sensor networks.

6.2 Feature Representation

The methodology is based on representation learning rather than mining of handcrafted statistical attributes. The processed signals windows are exactly passed to a one-dimensional convolutional neural network (1D-CNN), which is used to automatically recognize discriminative temporal features, such as abrupt amplitude changes, frequency and waveform morphology related to seismic activity.

This way, the feature engineering bias is minimized and generalization to the signal patterns that have not been seen is better.[3]

6.3 Model Training Strategy and Conversion

A stratified **50:50 split** was used to split the dataset into training and testing parts, with the earthquake and noise samples equally represented in both. The Adam optimizer was used to train the model because it has adaptive learning rate and converge quickly. The binary cross-entropy loss was used to measure error in classification in the binary detection task. The model was trained using **100 epochs and a batch size of 32.** Validation accuracy, loss were kept track of during training, in order to understand the behaviour of convergence and detect the event of overfitting. The model was trained, and then turned into **TensorFlow Lite format** to become smaller and less complex in terms of computation. The converted model was further converted into a **C header file** and made it possible to integrate into ESP32 firmware directly. Other optional optimization methods like post-training quantization were tested in order to lessen the use of memory without compromising on the accuracy.

6.4 Sensor Data Acquisition

The ESP32 microcontroller periodically (at rate of 50 Hz) samples the data of the acceleration information from the MPU6050 sensor via the I2C interface. The three-axis acceleration double-length vector magnitude is calculated to produce a one-dimensional vibration signal. Samples are received into a circular buffer of 200 values, which is a 4-second time window. When the buffer reaches capacity, the data is scaled and transduced to the neural network embedded to make an inference. The TensorFlow Lite Micro runtime runs the CNN model on the ESP32. The model gives a score of confidence between 0 and 1 which is the likelihood of seismic activity. A hysteresis mechanism based on cooldown was used to ensure a reduction in repeated alerts due to the residual vibrations. Once an event of detection has occurred, the system avoids additional warning messages during a specified period of time to maintain stability and augmented user experience. [1]

6.5 Experimental Validation

The system was tested in controlled laboratory environment using: Noisy surroundings of the environment, Simulation of vibration manually, Shaking situations of high intensity. Serial logs and LCD outputs were used to monitor system behaviour to determine the latency of detection, confidence stability, and the rate of false alarm.

7. Embedded Hardware Model Deployment.

This section demonstrates the use of the trained convolutional neural network on a resource-constrained microcontroller platform as well as the integration of the neural network with sensing and alerting hardware to detect earthquake in real-time mode.

7.1 Architecture Overview: Deployment.

The proposed system has an end-to-end deployment architecture that is depicted in figure 7.1. The prepared and optimized machine learning model is incorporated directly into the firmware of the ESP32 microcontroller and will perform the inference directly on the device without an external computing source or connection to the cloud.[5]

There are three functional layers of the deployment pipeline:

1. **Input Layer** Seismic vibrations detected by MPU6050 accelerometer and gyroscope module
2. **Processing Layer:** Real-time inference function implemented by ESP32 using TensorFlow Lite
3. **Output Layer:** Visual and Audio alerts produced using I2C Based LCD Display and Active Buzzer

7.2 Model Integration Process

The neural network model is then translated into TensorFlow Lite format and then converted to a C-compatible header file. This enables the model to be stored in flash memory of ESP32 and be directly accessed during running. The inference engine is embedded and during startup of the system it initializes the model and allocates a fixed-size tensor arena in the static memory.

7.3 Real-Time Data Flow

The working process of the implemented system is a continuous streaming paradigm:

1. The samples taken by the MPU6050 sensor are 3 axis acceleration samples with a constant sampling rate of 50 samples/second.
2. The ESP32 calculates the magnitude of an acceleration vector.
3. Samples are held on a circular buffer until a 4 second window (200 samples) is satisfied.
4. The buffered signal is made normalized and inferred in the embedded CNN model.
5. The model gives a score of confidence which is the probability of seismic activity.
6. In case the confidence surpasses a predetermined value, the system sends out visual and sound notifications.

7.4 Power and Communication Percentages

The system is powered with the help of a regulated 5V/3.3V supply to have compatibility with both

ESP32 microcontroller and peripheral components. Communication between ESP32 and the MPU6050 sensor along with the LCD display is done using I2C protocol using defunct SDA and SCL lines. This common bus design keeps wiring very simple and the total power usage is also very low and this means that the system can be deployed in the long term in distributed monitoring systems.

7.5 Alert and Feedback

The deployed system has two modes of feedback:

- **Visual Feedback:** A 16x2 LCD display is provided to show real-time confidence score and detection status.
- **Audio Feedback:** Active buzzer (connected to a digital feedback control on a microcontroller (a.k.a. a digital GPIO pin)) Buzzes for the user in certain high-confidence detection events.

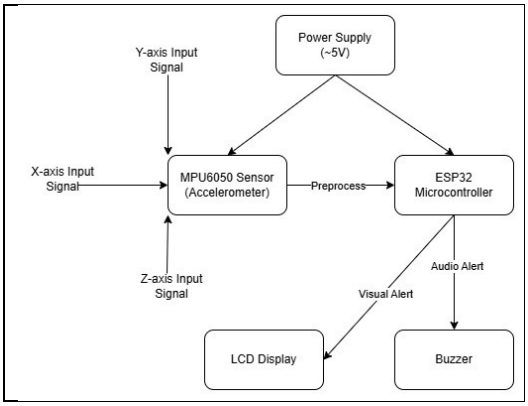

Figure 7.1: Proposed architecture of the TinyML-based Earthquake detection system and its embedded deployment. The vibration sensor (MPU6050) allows measuring the real-time vibration and sending it to the ESP32 microcontroller through the I2C interface. On-device inference is done on the trained CNN model the deployment of which is based on the TensorFlow Lite Micro.

8. Results

The content of this section involves the quantitative analysis of the proposed Earthquake detection system based on the TinyML platform and standard classification metrics and confusion matrix analysis. This is aimed at evaluating the capabilities of the system to effectively detect seismic events reliably, as well as to evaluate the strength of the system to seismic events in a real-time embedded system deployment environment.

8.1 Test Dataset Summary

The last assessment was done on a test set which had **21,784 time-series windows,** based on the processed seismic data. The distribution of the classes was as follows:

- **Earthquake (Class 1):** 19,284 samples
- **Background Noise (Class 0)**: 2500 samples

This distribution is based on a realistic deployment scenario where seismic events are not very frequent as compared to constant vibrations of the ground.

8.2 Confusion Matrix Analysis

A) Logistic Regression

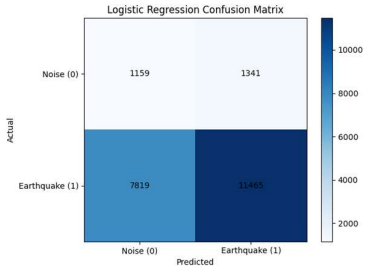

Figure 8.1: A confusion matrix of the classification of earthquake and noise for Logistic Regression.

	Predicted Noise (0)	**Predicted Earthquake (1)**
Actual Noise (0)	1159 (True Negatives)	1341 (False Positives)
Actual Earthquake (1)	7819 (False Negatives)	11465 (True Positives)

B) Random Forest

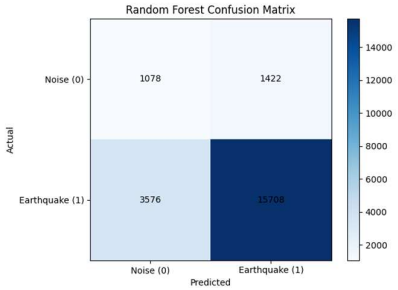

Figure 8.2: A confusion matrix of the classification of earthquake and noise for Random Forest.

	Predicted Noise (0)	**Predicted Earthquake (1)**
Actual Noise (0)	1078 (True Negatives)	1422 (False Positives)
Actual Earthquake (1)	3576 (False Negatives)	15708 (True Positives)

C) 1-D CNN (Proposed Model):

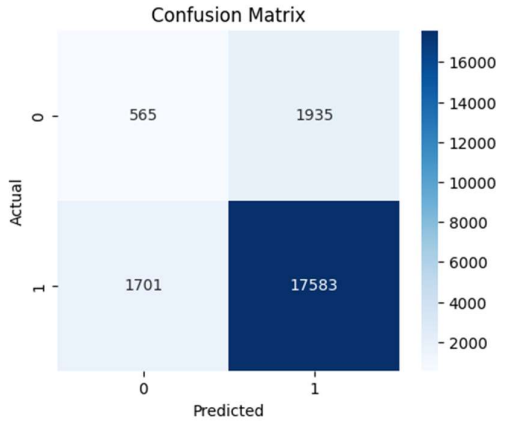

Figure 8.3: A confusion matrix of the proposed CNN classification of earthquake and noise.

	Predicted Noise (0)	**Predicted Earthquake (1)**
Actual Noise (0)	565 (True Negatives)	1,935 (False Positives)
Actual Earthquake (1)	1,701 (False Negatives)	17,583 (True Positives)

The model has high rate of true positives which signify high sensitivity to seismic activity even when it is trained on a smaller set of data. Nevertheless, there are still significant false positives indicating the difficulty of differentiating between the noise of high amplitude and actual seismic activities.

8.3 Metrics of Performance Evaluation.

Several measurements were applied to give a profound analysis:

1. **Accuracy:** Accuracy of classification overall
2. **Precision:** Percentage of the correctly predicted earthquakes of all the predicted earthquakes.
3. **Recall:** Percentage of successfully identified earthquakes of all real earthquake samples.
4. **F1-Score:** Precision and Recall Harmonic Mean
5. **Confusion Matrix:** Display of True Positives, False Positives, True Negatives, False Negatives

The high priority assigned to recall was because of the safety critical nature of the earthquake detection where false alarm does not cost as much as a missed true event.

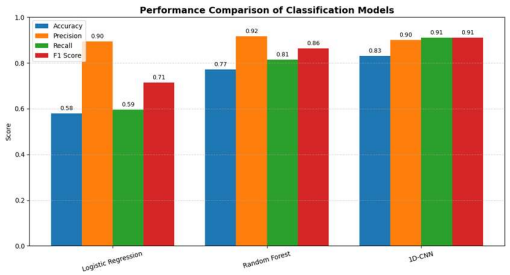

Figure 8.4: Performance Comparison of Classification Models

Measures of Quantitative Performance.

Evaluation of classification performance was based on precision, recall, F1-score and accuracy. Table 8.1 summarises the findings.

Table 8.1: Performance Metrics

Model	**Accuracy**	**F1-Score**	**Precision**	**Recall**
Logistic Regression	0.58	0.71	0.89	0.59
Random Forest	0.77	0.86	0.91	0.82
1-D CNN (Proposed Model)	0.83	0.91	0.90	0.91

8.4 Interpretation of Results

The model was able to obtain a **recall of 0.91 on the earthquake class**, which implies that more than 91 percent of real seismic events were identified in spite of the smaller training set size. This shows the strength of the trained time-related features and the appropriateness of the architecture to be used in safety-related detection scenarios.

The **precision of 0.90** also shows that most of the predicted earthquake occurrences are appropriate seismic activity, which proves the accuracy of the alert generation under limited training measures. The **recall of 0.23** was observed in the noise class, which implies that about 23 percent of background noise windows were correctly identified. Most noise samples were mistaken to be earthquakes which made this sample have a high rate of false-positive. This behaviour represents a trade-off in the design to either pay more attention to seismic events with a sacrifice in the ability to reject noise. This is a type of trade-off that can be sometimes tolerated in early-warning and disaster mitigation systems because false alarms are usually not so critical, compared to missed detections.

The total accuracy also went down, by a small margin of about **86 to 83 percent,** when compared to the **80:20 split test** and therefore the model still exhibits good generalization power even using a much smaller **training set (50:50).** The consistency of earthquake-type recall in both architectures indicates that the convolutional architecture is

effective to learn general seismic patterns instead of memorizing its particularity to the datasets.[4]

8.5 Embedded Deployment validation

The trained and optimized model was able to be deployed on the ESP32 platform with TensorFlow Lite Micro. Continuous accelerometer data were sampled at 50Hz and inferred on a sliding spectrum of 4 seconds with real time. [3], [7]

During hardware testing:

- A high occurrence was always noted in high confidence outputs when there was occurrence of strong vibration.
- Although the ambient environmental conditions had low confidence values.
- The hysteresis mechanism based on cooldown minimized the number of repeat alerts due to the residual vibrations.

These observations validate the fact that the performance of the model offline can be successfully applied to the real-world scenario of inference on the device.

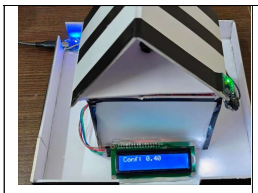

Fig 8.5 System correctly rejecting background vibrations (Low Confidence 0.40) without triggering a false alarm.

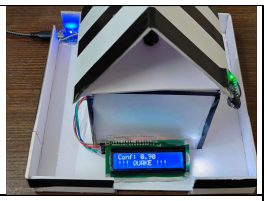

Fig 8.6 System correctly detecting a simulated earthquake with high confidence (0.90) and triggering the visual alert.

8.6 Summary of Findings

The results of the experiment prove that the suggested TinyML-based system:

- Classifies approximately 83 percent on actual seismic windows in the real world.
- Receives high sensitivity (91% recall) for earthquake detection
- Works successfully in real-time on a low power ESP32 micro controller
- Stable confidence-based alerting with inbuilt hysteresis control

These results confirm the practicability of the implementation of seismic detection models based on the application of deep learning and implemented on the hardware of microcontroller sizes to scale and responsive seismic monitoring systems.[1], [3], [7]

Conclusion

In this study, the design, implementation, and evaluation of a TinyML-powered earthquake detector on a seismic event real-time monitoring on an IoT platform were presented with low power usage. The paper shows that deep learning-based intelligent seismic classification can be successfully implemented at the edge without the use of cloud-based infrastructure. The work demonstrates the viability of real-time, decentralized earthquake detection by taking seismic signal processing, convolutional neural networks, and embedded systems engineering as a single architecture. A step-by-step pipeline was created that involves data set acquisition, signal preprocess, CNN model training, and model optimization to be used on a microcontroller. Sliding window segmentation, downsampling, and per-window normalization data augmentation methods were using to create standardized time-series features that are closely related to real-world streaming environment conditions. A compact 1D-CNN was developed with the particular focus on resource-constrained hardware and the compatibility with TensorFlow Lite Micro. This trained framework was effectively ported and implemented on ESP32 microcontroller and allowed real-time inference with 4-second sliding windows at 50 Hz.

The suggested 1D-CNN model showed a high level of classification, significantly high recall on earthquake events under both standard (80:20) and more stringent (50:50) train-test splits. The focus on recall is in line with safety critical traits of seismic monitoring where reduced misdetections are more significant than reduced false alarms. The validation of hardware ensured that the deployed system gives high confidence levels during earthquake episodes simulated and lower confidence levels during ambient vibrations. The findings confirm the seismic classification using deep learning can be effectively implemented using edge computing systems that operate on low-power and in real-time. Future research will aim at enhancing noise discrimination based on adaptive thresholding, feature integration in the frequency domain, and data fusion between multiple sensors. Further improvements can be geolocation via GPS, and alert dissemination which in turn is linked to the cloud, and massive implementation of interconnected sensor nodes to create distributed seismic monitoring networks. These developments would provide the ability to have scalable, cost-efficient, and autonomous seismic sensing systems that would be applicable to smart buildings, college, and resource-limited settings.

References

[1] Warden, P. and Situnayake, D., TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers, O'Reilly Media, 29 Mal, 2019.

[2] Allen, R. M., & Melgar, D. (2019). *Earthquake Early Warning: Advances, Scientific Challenges, and Societal Needs*. https://doi.org/10.1146/annurev-earth-053018

[3] Sharma, A., & Nanda, S. J. (2025). Spatial-temporal seismicity analysis using TSOM and variational density peak clustering. *Environmental and Ecological Statistics*, *32*(1), 229–281. https://doi.org/10.1007/s10651-024-00641-7

[4] Yoon, C. E., O'Reilly, O., Bergen, K. J., & Beroza, G. C. (2015). Earthquake detection through computationally efficient similarity search. *Science Advances*. https://doi.org/10.1126/sciadv.1501057

[5] Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., & Wassermann, J. (2009). An XML-SEED Format for the Exchange of Synthetic Seismograms. http://numpy.scipy.org

[6] *ESP32 Series Datasheet Version 5.1 2.4 GHz Wi-Fi + Bluetooth ® + Bluetooth LE SoC Including*. (n.d.). www.espressif.com

[7] Banbury, C., Zhou, C., Fedorov, I., Navarro, R. M., Thakker, U., Gope, D., Reddi, V. J., Mattina, M., & Whatmough, P. N. (2021). *MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers*. http://arxiv.org/abs/2010.11267

[8] Albericio, J., Judd, P., Hetherington, T., Aamodt, T., Jerger, N. E., & Moshovos, A. (2016). Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing. *Proceedings - 2016 43rd International Symposium on Computer Architecture, ISCA 2016*, 1–13. https://doi.org/10.1109/ISCA.2016.11

[9] Arduino, "ESP32 Technical Reference Manual," Espressif Systems, 2023, pg. https://docs.espressif.com

[10] TensorFlow Team, TensorFlow Lite for Microcontrollers: An Overview # TensorFlow AI. https://www.tensorflow.org/lite/microcontrollers

[11] ben dor, guy (2023), "ISN seismic waveforms", Mendeley Data, V1, doi: 10.17632/k6hpr8m3f2.1 https://data.mendeley.com

[12] U.S. Geological Survey (USGS), "Earthquake Detection and Monitoring Systems," 2022. https://earthquake.usgs.gov

[13] Lane, N. D., Bhattacharya, S., Mathur, A., et al., "Squeezing Deep Learning into Mobile and Embedded Devices" IEEE Pervasive Computing, vol. 16, no. 3, pp. 82-88, 2017.

[14] Chollet, F., Keras: The Python Deep Learning Library, Journal of Machine Learning Research, vol. 19, no. 1, pp. 1-9, 2018.

[15] ImageNet Classification with Deep Convolutional Neural Networks. Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012, Advances in Neural information Processing Systems (Neurips).

[16] TensorFlow Team, "Post-Training Quantization for TensorFlow Lite" Google AI, 2023. https://www.tensorflow.org/lite/performance/posttrainingquantization

[17] Saputra, M. R. U., et al., "Earthquake Detection with tinyML," Seismological Research Letters, vol. 94, no. 4, pp. 1–13, Apr. 2023.

[18] Al-Saadi, F. Z. R., et al., "Earthquake Early Warning Using Low-Cost MEMS Sensors," IEEE Internet of Things Journal, vol. 8, no. 4, pp. 2248–2257, 2021.

[19] Perol, T., Gharbi, M., & Denolle, M., "Convolutional neural network for earthquake detection and location," Science Advances, vol. 4, no. 2,p. e1700578, Feb. 2018.

[20] Magana-Zook, S. A., Clayton, R. W., & Cochran, E. S., "Seismic Event Classification in machine learning for Low Power Sensor Networks," IEEE Sensors Journal, vol. 21, no. 10, pp. 11843–11852, May 2021.