International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

ForestGuard: A TinyML-Based Acoustic Surveillance System for Intelligent Forest Monitoring

Chinchu Paulose[1] ,Saraung Babu[2] ,Sradha TR[3] ,Deepak D Nair[4] ,Safa Sajith C S[5]

[1]Asst Professor, Dept of CSE , Sree Narayana Gurukulam College of Engineering, Kochi, India
chinchup@sngce.ac.in
[2]Student, Dept of CSE, Sree Narayana Gurukulam College of Engineering, Kochi, India
saraungbabu@gmail.com
[3]Student, Dept of CSE, Sree Narayana Gurukulam College of Engineering, Kochi, India
sradhatr8@gmail.com
[4]Student, Dept of CSE, Sree Narayana Gurukulam College of Engineering, Kochi, India
deepakdnair07@gmail.com
[5]Student, Dept of CSE, Sree Narayana Gurukulam College of Engineering, Kochi, India
safasajith1@gmail.com

Abstract - Forests are increasingly threatened by illegal logging, wildlife poaching, forest fires, and unauthorized human intrusion. Conventional surveillance approaches such as manual patrolling, camera traps, and satellite monitoring are limited by high operational costs, restricted coverage, delayed response times, and dependency on visibility conditions. Acoustic monitoring presents a promising alternative, as many harmful forest activities produce distinctive sound signatures.

This paper presents ForestGuard, a complete end-to-end TinyML-based acoustic surveillance system for intelligent forest monitoring. The proposed system employs an ESP32 microcontroller integrated with a digital I2S microphone to perform real-time on-device sound classification using a quantized deep learning model. The classifier detects chainsaw activity, gunshots, elephant vocalizations, lion/tiger vocalizations, fire crackling sounds, and human screams, along with an additional "unknown" class to handle environmental background noise.

The deployed system achieved 85.2% validation accuracy and 70.53% real-world testing accuracy, with an AUC of 0.95, while maintaining a compact model size of approximately 69 KB suitable for embedded deployment. The architecture integrates a Spring Boot backend server and a React Native mobile application for real-time alert visualization and scalable multi-device monitoring.
The results demonstrate the feasibility of scalable, low-cost, edge-based acoustic surveillance for forest protection and wildlife conservation

Key Words: TinyML, Acoustic Surveillance, ESP32, Edge AI, Forest Monitoring, IoT

1.INTRODUCTION

Forests play a vital role in sustaining biodiversity, regulating climate systems, and maintaining ecological balance. However, illegal logging, wildlife poaching, forest fires, and unauthorized human activities continue to threaten forest ecosystems globally. In countries such as India, vast and remote forest areas pose significant challenges for continuous monitoring using conventional surveillance techniques.

Traditional forest surveillance methods rely on human patrols, satellite imagery, and camera-based systems. These approaches are often constrained by high maintenance costs, limited coverage area, lighting dependency, and delayed detection capabilities. Many illegal activities occur during nighttime or in dense vegetation, where visual monitoring becomes ineffective.

Acoustic monitoring provides an alternative approach since numerous forest disturbances generate characteristic sound signatures. Chainsaw operations, gunshots, wildlife distress calls, fire crackling, and human screams can be detected even under low-visibility conditions.

Recent advancements in Tiny Machine Learning (TinyML) enable the deployment of neural networks directly on microcontrollers with constrained memory and processing resources. This allows real-time inference without continuous cloud connectivity. ForestGuard leverages TinyML principles to design a decentralized acoustic surveillance system capable of autonomous edge-based detection and real-time alerting.

2. RELATED WORKS

Acoustic monitoring has been widely explored in environmental and wildlife research. Early approaches focused on recording and storing large volumes of audio data for offline analysis. While effective for ecological studies, these systems are unsuitable for real-time surveillance due to storage and energy constraints.

Recent studies have applied machine learning techniques to classify environmental sounds, particularly for urban noise monitoring and wildlife detection. Publicly available datasets such as ESC-50 and UrbanSound8K have facilitated the development of sound event classification models. However, many of these models are computationally intensive and designed for cloud or desktop environments.

With the emergence of TinyML, researchers have demonstrated that lightweight neural networks can be deployed

on microcontrollers for tasks such as keyword spotting, anomaly detection, and basic audio classification. Some works have explored bird call detection and animal sound recognition using embedded devices. Despite these advances, many existing systems either focus on a narrow set of sound classes or rely on external processing units.

ForestGuard differs from existing approaches by emphasizing complete on-device inference using a single embedded platform. The system is designed specifically for forest surveillance scenarios, addressing challenges such as background noise, limited connectivity, and power efficiency. The inclusion of an "unknown" class further enhances robustness by reducing false detections caused by non-critical ambient sounds.

3. PROPOSED SYSTEM

ForestGuard is designed as an autonomous acoustic sensing unit capable of continuous operation in forest environments. The system captures ambient audio signals, processes them locally, and classifies sound events of interest using a TinyML model.

3.1 System Overview

The core components of the system include:

- ESP32 microcontroller
- Digital microphone interfaced through I2S/PDM
- TinyML-based sound classification model
- Local decision logic for event identification

Audio signals are sampled at a fixed frequency and segmented into short time windows. Each audio segment is passed to the embedded classifier, which produces probability scores for each predefined sound class. Based on these scores, the system determines whether a critical event has occurred.

3.2 Sound Categories

The system is trained to recognize the following sound classes:

- Chainsaw
- Gunshot
- Elephant vocalizations
- Lion or tiger vocalizations
- Fire crackling sounds
- Human scream
- Unknown (background and non-relevant sounds)

The "unknown" category plays a significant role in handling environmental noise such as wind, rain, insects, and distant sounds, thereby improving overall system reliability

4. SYSTEM ARCHITECTURE

The ForestGuard architecture follows a decentralized edge-computing model. All audio processing and inference are performed locally on the embedded device, eliminating the need for continuous data transmission.

4.1 Hardware Architecture

The ESP32 microcontroller serves as the central processing unit due to its low power consumption, sufficient memory, and support for digital audio interfaces. A compact digital microphone is used to capture ambient sound with minimal noise interference.

In addition, the selected hardware components are optimized for continuous operation in resource-constrained environments. The ESP32 supports low-power operating modes, which can be utilized to reduce energy consumption during idle periods, making the system suitable for long-term deployment in remote forest locations. The use of a digital microphone eliminates the need for complex analog signal conditioning, improving noise immunity and simplifying the hardware design. This combination of low-power processing and efficient audio acquisition enables reliable acoustic monitoring while maintaining a compact and cost-effective hardware footprint.

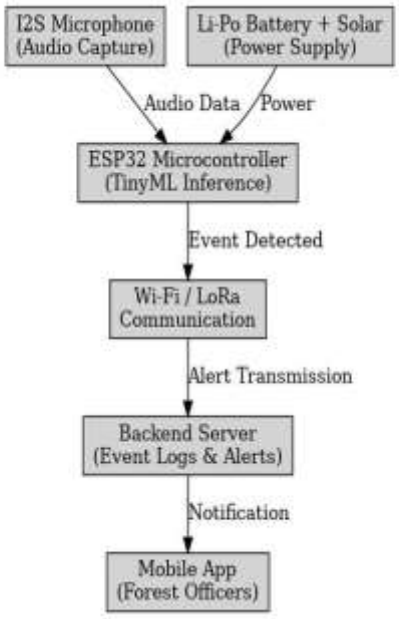

Fig. 1: Hardware Architecture

4.2 Software Workflow

1. Continuous audio capture
2. Segmentation into fixed-length windows
3. MFCC feature extraction
4. Neural network inference
5. Probability score generation
6. Temporal smoothing (2 of last 4 windows rule, confidence ≥ 60%)
7. Alert triggering

The workflow ensures reliable detection while minimizing false positives.

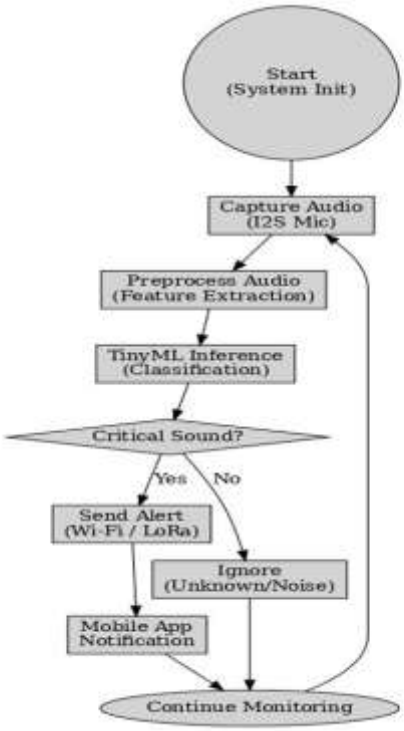

Fig. 2: System Workflow

4.3 Backend and Mobile Architecture

ForestGuard incorporates a centralized backend infrastructure to enable scalable event aggregation, storage, and visualization. The backend is implemented using the Spring Boot framework and follows a REST-based architecture.
The backend exposes RESTful APIs for:

- Device registration and identification
- Event logging and alert reception
- Persistent storage of classified events
- Retrieval of historical event records
- Device-wise monitoring and filtering

Each ESP32 node transmits alert events via HTTP POST requests to the backend server upon satisfying classification confidence (≥ 60%) and temporal smoothing conditions. The transmitted payload includes predicted class label, timestamp, and device identifier. Upon receiving the request, the backend validates and stores the event in a structured database.
The communication workflow between edge device, backend server, database, and mobile application is illustrated in Fig. 3 (Backend Communication Flow).
The React Native mobile application functions as the monitoring interface for forest authorities. It retrieves data from the backend through secure REST APIs and provides:

- Real-time alert notifications
- Device-wise event categorization
- Historical event tracking
- Map-based visualization of multiple ESP32 nodes

To support large-scale deployment, the system architecture allows multiple sensing nodes to communicate with a centralized backend. Each device is uniquely identified, enabling simultaneous monitoring of geographically distributed forest regions. The scalable deployment architecture is illustrated in Fig. 4 (Multi-Device Deployment Architecture).
This design ensures separation of edge inference and centralized monitoring while maintaining low communication overhead and real-time responsiveness.

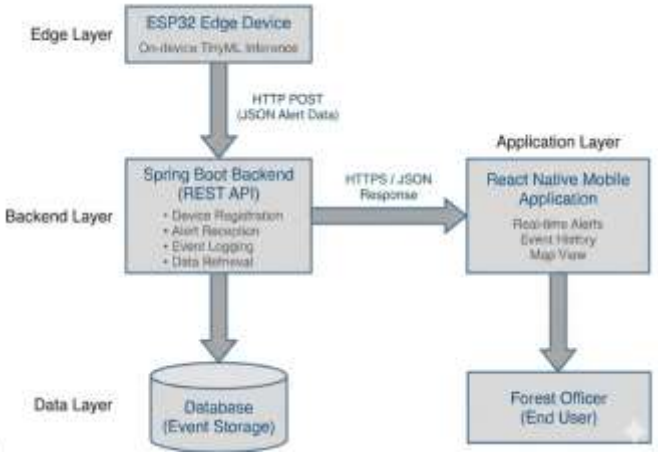

Fig. 3: Backend Communication Flow

Fig. 4: Multi-Device Deployment Architecture

5. RESULTS AND DISCUSSION

The TinyML classifier was trained using Mel-Frequency Cepstral Coefficient (MFCC) features and a quantized int8 neural network architecture optimized for embedded deployment. The model was trained and validated using the Edge Impulse platform.
During validation, the classifier achieved an overall accuracy of **85.2%** with a loss value of **0.52**, indicating effective convergence without significant overfitting. The weighted performance metrics further demonstrate balanced classification capability across all sound categories.
The validation performance metrics are summarized in Table 1.

Table 1. Validation Performance Metrics

Metric	Value
Validation Accuracy	85.2%
Validation Loss	0.52
Weighted Precision	0.86
Weighted Recall	0.82
Weighted F1 Score	0.83
Area Under ROC Curve (AUC)	0.95

The Area Under the Receiver Operating Characteristic Curve (AUC) of 0.95 indicates strong separability between sound classes and robust discriminative capability of the trained model.

5.2 Testing Performance

To evaluate deployment feasibility, the trained TinyML model was tested on unseen data under real-world operating conditions. During testing, the system achieved an overall accuracy of 70.53%.

The reduction from validation accuracy (85.2%) to testing accuracy (70.53%) reflects the impact of environmental variability and acoustic complexity present in real deployment scenarios. Contributing factors include background noise, overlapping frequency characteristics between certain sound classes (such as fire and chainsaw), and variations in recording conditions.

Despite these challenges, the classifier demonstrated reliable detection of critical events. The implementation of a confidence threshold (≥ 60%) combined with temporal smoothing logic (event triggered if detected in 2 of the last 4 windows) helped reduce false positives and improve operational robustness.

A detailed per-class performance analysis based on the confusion matrix is presented in Section 5.3.

5.3 Per-Class Analysis

The confusion matrix shown in Fig. 5 illustrates the class-wise performance of the deployed model under testing conditions.

Strong classification performance was observed for gunshot (90.5% accuracy, F1 score 0.95), indicating clear acoustic separability. The elephant, lion/tiger, and scream classes demonstrated moderate to strong recognition performance with acceptable F1 scores.

The unknown class achieved reliable performance (F1 score 0.83), confirming its role in reducing false detections from ambient forest noise.

Lower accuracy was observed for chainsaw and fire classes. These sounds exhibited misclassification patterns, partly due to overlapping spectral characteristics and environmental variability during testing.

Overall, the confusion matrix indicates that impulsive and distinct acoustic events are detected reliably, while continuous or spectrally similar sounds present greater classification challenges.

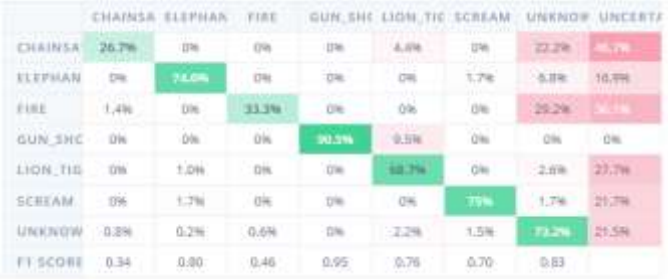

Fig. 5: Confusion matrix of the deployed classifier

5.4 End-to-End Validation

The complete ForestGuard system pipeline was successfully validated under deployment conditions. The operational flow consisted of:

ESP32 Edge Device → Spring Boot Backend → Database → React Native Mobile Application

When classification confidence exceeded the defined threshold (≥ 60%) and the temporal smoothing condition (2 of the last 4 windows) was satisfied, an alert event was generated on the ESP32 device. The event data, including predicted class label and timestamp, was transmitted to the backend server via HTTP.

Upon receiving the alert, the backend stored the event in the database and made it available to the mobile application through REST APIs. The React Native application displayed the alert in real time, including device-wise monitoring and map-based visualization for multi-node deployment.

The embedded model operated within the memory and computational constraints of the ESP32, with inference performed in near real time on 1-second audio windows. No memory overflow or runtime instability was observed during continuous operation.

The inclusion of the "unknown" class, combined with confidence thresholding and temporal smoothing, significantly reduced false positives during testing. The successful integration of edge inference, backend processing, and mobile visualization confirms the practical feasibility of the proposed system for scalable forest monitoring.

The communication workflow is illustrated in Fig. 3.

6. LIMITATIONS

- Performance degradation under high wind and heavy rain
- Acoustic similarity between fire and chainsaw
- Limited forest-specific dataset diversity
- Semi-controlled testing environments

These factors provide opportunities for future improvement.

7. FUTURE SCOPE

The ForestGuard system can be further enhanced in several ways:

- LoRa-based dual ESP32 relay architecture
- Solar-powered autonomous deployment
- Federated learning for distributed retraining
- Larger forest acoustic dataset collection
- Adaptive threshold optimization

- Integration with GIS-based forest monitoring systems
- Edge anomaly detection models

8. CONCLUSIONS

ForestGuard demonstrates the practical feasibility of deploying a TinyML-based acoustic surveillance system for intelligent forest monitoring. By performing inference directly on an ESP32 microcontroller, the system eliminates reliance on continuous cloud processing while maintaining real-time responsiveness.

With 85.2% validation accuracy and 70.53% real-world testing accuracy, the system proves effective for detecting critical forest events. The integration of backend infrastructure and mobile monitoring enhances scalability and operational usability.

ForestGuard highlights the potential of edge AI technologies in supporting environmental protection and wildlife conservation initiatives.

.

REFERENCES

1. Sabbella, H. R., Nair, A. R., Gumme, V., Yadav, S. S., Chakrabartty, S., and Thakur, C. S., "An Always-On TinyML Acoustic Classifier for Ecological Applications," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2634–2638, 2022.
2. Roy, B. B., Das, S., and Mondal, U. K., "TinyML-Driven Sensor Nodes for Energy-Efficient Acoustic Event Detection in Pervasive Acoustic Wireless Sensor Networks," *Journal of Telecommunications and Information Technology*, vol. 2, no. 2, pp. 1–10, 2025.
3. Ayankoso, S., Wang, Z., Shi, D., Yang, W., Vikiru, A., Kamau, S., Muchiri, H., and Gu, F., "Development of Long-Range, Low-Powered and Smart IoT Device for Detecting Illegal Logging in Forests," *Journal of Dynamics, Monitoring and Diagnostics*, vol. 3, no. 3, pp. 190–198, 2024.
4. Rajesh, M., Wadhwa, T., Cherukuri, A. K., Kamalov, F., Jonnalagadda, A., and Ray, S., "Preventing Illegal Deforestation Using Acoustic Surveillance," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 16, no. 1, pp. 70–86, 2024.
5. Mporas, I., Perikos, I., Kelefouras, V., and Paraskevas, M., "Illegal Logging Detection Based on Acoustic Surveillance of Forests," *Applied Sciences*, vol. 10, no. 20, Article 7379, 2020.
6. Khanal, K., Rana, S., Ghimire, U. K., and Neupane, A., "A Low-Power IoT Architecture for Real-Time Detection and Notification of Unauthorized Tree Poaching," *Sensors*, vol. 24, no. 11, Article 3432, 2024.
7. Sheng, Z., Pfersich, S., Eldridge, A., Zhou, J., Tian, D., and Leung, V. C. M., "Wireless Acoustic Sensor Networks and Edge Computing for Rapid Acoustic Monitoring," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 64–74, 2019.
8. Mainetti, L., "Acoustic Identification of Wood-Boring Insects with TinyML," *IEEE Access*, vol. 11, pp. 110963–110972, 2023.
9. Kumar, A., Reddy, S., and Sharma, P., "AI and IoT-Based Smart Surveillance for Wildlife Protection," *Journal of Advances in Computer Science and Technology*, vol. 12, no. 4, pp. 1–15, 2025.
10. Wrege, P. H., Rowland, E. D., Keen, S., and Shiu, Y., "Acoustic Monitoring for Conservation in Tropical Forests: Examples from Forest Elephants," *Biotropica*, vol. 49, no. 6, pp. 888–897, 2017.