2026 Volume: 02 Issue: 11 www.ijrpa.com ISSN 2456-9995 Research Article

International Journal Research Publication Analysis

Page: 01-12

ABHAYA: IMPLEMENTATION OF AN ON-DEVICE AUDIO-ASSISTED PERSONAL SAFETY APPLICATION WITH MUFFLED-SPEECH DETECTION

Rekha V.,*[1] Abhishek A.,[2] Maanya Vittal B. G.,[2] Samartha B. S.,[2]

[1]Assistant Professor, Dept. of CSE, Jyothy Institute of Technology, Karnataka, India.

[2]Student, Dept. of CSE, Jyothy Institute of Technology, Karnataka, India.

Article Received: 24 September 2026

Article Revised: 14 October 2026

Published on: 03 November 2026

***Corresponding Author: Rekha V.**

Assistant Professor, Dept. of CSE, Jyothy Institute of Technology, Karnataka, India.

DOI: https://doi-doi.org/101555/ijrpa.4306

ABSTRACT

Personal safety applications commonly require users to initiate a panic action, which may be difficult during an emergency. This paper describes the documented implementation of Abhaya, an Android-first application developed in Flutter and Dart that combines a manual SOS interface, locally stored trusted contacts, device-based location retrieval, an audible siren, and a microphone-driven distress detection pipeline. Audio is sampled at 16 kHz and passed to an on-device TensorFlow Lite convolutional classifier through a detector abstraction. A stub-first architecture separates testing of the alert workflow from model development. The proposed machine-learning experiment compares conventional training with low-pass filtering, additive noise, and amplitude reduction to approximate muffled audio. The project report records 25 passing functional/integration cases and lists muffled-distress recall values of 72.4% and 89.6% for baseline and augmented models; however, the corresponding experiment artifacts and evaluation protocol are absent, so these values are treated as unverified report claims rather than validated research results. The paper delineates implemented functions, planned extensions, and reproducibility requirements.

INDEX TERMS: Convolutional Neural Network, Distress Detection, Flutter, Muffled Speech, Tensorflow Lite, Personal Safety.

I. INTRODUCTION

In threatening situations, the physical action required to unlock a phone and select an emergency function may be impractical. Abhaya is documented as a mobile safety prototype that brings conventional assistance functions together with a local audio-classification path. Its central engineering aim is to maintain a consistent response workflow regardless of whether an alert begins from the SOS interface or an audio detection event.

This problem is a systems-engineering challenge as well as an audio-classification problem. A useful detector must provide a dependable path from a microphone observation to an understandable, user-controlled alert action. In particular, a probabilistic model output does not independently verify an emergency: the application must coordinate permissions, repeated observations, notification presentation, location availability, and appropriate failure handling. The implementation described here therefore treats the detector as one component of a larger safety workflow rather than an autonomous proof of danger.

The engineering question addressed by this manuscript is how to construct and document a modular mobile architecture that permits an audio classifier to be integrated and replaced while keeping the contact-management and emergency-alert services stable. A related empirical question is whether muffling augmentation increases detection recall under acoustically obstructed test conditions. The project report provides indicative values for the latter question, but lacks the evaluation artifacts needed for an independent assessment; this distinction guides the interpretation of the results below.

This article focuses on the implemented mobile architecture and the documented model interface rather than claiming a fully deployed emergency-response service. The main engineering contributions described in the project report are a modular detector interface, a shared alert service, continuous in-app microphone processing, and a controlled stub detector for testing alert propagation independently of recognition accuracy.

II. RELATED WORK

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), described by Livingstone and Russo [1], provides acted emotional speech recordings that can support an initial emotion-related audio classification experiment. It does not by itself constitute a real-world emergency-distress dataset, and the mapping of fearful and angry speech to “distress” is a task-specific assumption. The Abhaya report uses these emotions as positive examples and neutral, calm, and happy as negative examples. Relevance to whispered or obstructed real-world emergency audio therefore requires separate validation.

Flutter provides a shared application framework for mobile development [2]. Abhaya uses this stack for Android-first interfaces and isolates audio inference using TensorFlow Lite. Porcupine provides a Flutter wake-word integration path [3], but the source explicitly defers this integration. The project specifies a CNN with a mel-spectrogram front end, followed by convolutional and classification layers, but does not supply the final layer configuration or all training hyperparameters.

Research on acoustic scene understanding and speech-based emotion recognition motivates the use of time-frequency features and neural classifiers, but the transfer from a curated recording collection to naturally occurring emergencies is not established by those methods alone. The source project uses a set of acted RAVDESS emotions as operational labels for distress. Accordingly, findings based on these labels should be interpreted as results on the defined audio-classification task rather than as a field-tested claim about preventing harm.

Abhaya's distinguishing implementation concern is the integration of inference with a controllable emergency workflow on a phone.

A further distinction is between wake-word spotting and distress-sound classification. The former identifies an intentionally spoken activation term, while the latter attempts to infer a state from short audio segments. Both appear in the wider project objectives, but the supplied implementation description establishes the more detailed interface for distress classification. Wake-word support is therefore discussed as a deferred extension, not as part of the empirically evaluated classifier.

III. PROPOSED METHODOLOGY AND SYSTEM ARCHITECTURE

A. Research objective and dataset

The objective is to build a modular, device-side distress detection and emergency-assistance workflow, and to evaluate whether muffling-aware augmentation improves the recall of distress-related audio compared with an otherwise comparable baseline. The report identifies the RAVDESS dataset [1] as the training source and maps fearful and angry speech to the positive class and neutral, calm, and happy speech to the negative class. RAVDESS records enacted vocal emotion rather than real emergency events; consequently this class definition is a proxy. Neither the number of selected clips nor the training, validation, and testing proportions are consistently evidenced in the supplied report.

B. System flow and architecture

The intended system design has four connected paths: microphone capture and local audio inference; manual SOS activation; a shared emergency alert service; and trusted-contact and

device interfaces. For both manual and detector-initiated events, the application prepares an emergency action through the common alert mechanism. Fig. 1 summarizes the flow represented in the implementation description. This is a schematic synthesized from the report, not an image of a deployed device.

The architecture separates responsibilities to limit the effect of component failures. The microphone acquisition component should be permitted to stop or reject permission without corrupting stored contacts; conversely, initiating a manual SOS should not depend on whether the audio classifier is loaded.

Both triggering paths converge at a shared alert interface to reduce duplicated location, notification, and cancellation logic. This arrangement also permits deterministic testing of alert behavior by substituting a stub detector for the learned model.

The reported design uses a user-facing countdown prior to preparation of an SMS action. Importantly, invoking an SMS URI is not identical to confirming delivery to a recipient. A research evaluation must separate successful generation of a draft, user approval of the device messaging interface, transmission, and receipt. The present report supports discussion of the application-level action but does not demonstrate end-to-end delivery reliability under adverse network conditions.

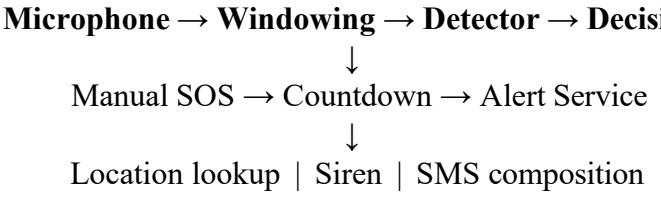


Fig. 1. Report-derived logical emergency activation workflow.

IV. IMPLEMENTATION

A. Application modules

The documented implementation comprises a Safety Circle, distress detection, alert and device services, and supporting interfaces. TrustedContact objects hold names and telephone numbers; ContactService serializes entries as JSON in shared_preferences under abhaya_trusted_contacts. Duplicate phone numbers are rejected, and contacts can be added and removed. The current report describes local persistence rather than a deployed cloud contact database.

The safety-circle subsystem exposes list management as a local service instead of coupling contact records to the alert user interface. As described in the project report, each trusted

contact stores a name and telephone number and can be serialized to JSON for persistence in shared_preferences. Adding an existing phone number is rejected, and a removal request is checked against the current list bounds. These details matter because duplicates could produce repeated attempted notifications, while an invalid index must not remove an unintended contact.

The report also describes auxiliary features such as a fake-call interaction and audio or vibration feedback. These may offer additional user controls, but their presence must not be conflated with empirical verification of emergency effectiveness. In particular, a feature can function as designed in a software test without demonstrating that it will improve safety outcomes in uncontrolled real-world situations.

AlertService provides a canonical activation path (triggerAlert() and dispatchAlert()). It retrieves available location data with geolocator and builds a Google Maps link, controls an audible siren through audioplayers, and prepares a contact alert using a device SMS URI. Importantly, the report states that the SMS application is opened with a prefilled message and the user must confirm sending; it does not demonstrate silent automatic delivery. A fake-call interface, ringtone, and vibration provide a separate preventive interaction.

B. Audio acquisition and scoring

The detailed code-oriented description specifies a 16-kHz mono, 16-bit PCM audio stream acquired using the record package. ListeningService requests microphone permission, converts samples to normalized floating-point values, buffers samples into windows, and passes each window to a DistressDetector implementation. TfliteDistressDetector loads the CNN .tflite asset via tflite_flutter, while StubDistressDetector provides deterministic simulated outputs for integration testing.

For integration reproducibility, the number of waveform samples must agree with the model interpreter's input tensor. At a 16,000 Hz sample rate, two seconds correspond to 32,000 samples and three seconds to 48,000. Passing the wrong length may cause an inference failure or require trimming or padding; any such transformation changes the observed acoustic context. The two documented configurations cannot be treated as interchangeable without inspecting the exported tensor signature and the audio-buffering source code.

Practical deployment also depends on resource and lifecycle behavior. A foreground-only demonstration is not equivalent to continuous background monitoring: mobile operating systems may pause microphone acquisition, restrict permissions, or stop an application after a lifecycle transition. A deployment study would need to record battery drain, memory consumption, inference latency, and behavior during incoming calls, permission revocation,

and repeated start–stop cycles. The uploaded report does not provide such measurements, so no runtime performance claim is made here.

The report is inconsistent about the exact input length: its model contract states [1, 48000] (three seconds), whereas its later implementation account states 32,000 samples (two seconds). The later description explicitly refers to the currently supplied detector code, but no executable repository or exported model was provided for independent inspection. The precise deployed window length must be confirmed from the final model tensor metadata before publishing an experimental comparison.

C. *Decision and alert flow*

The inferred scalar score is compared with a configured probability threshold and a consecutive-hit requirement. The report variously identifies 0.50 and 0.64 as the threshold in different descriptions; the code-oriented section specifically gives 0.64 as the constructor default while also describing a home-screen configuration of 0.50. Thus, no single verified production threshold can be asserted. A 30-second alert cooldown is described to reduce repeated triggering. The audio callback enters a countdown/alert path rather than unconditionally notifying contacts.

Fig. 1 expresses the source-described workflow: microphone permission and streaming → waveform buffering → detector scoring → threshold and consecutive-hit decision → countdown and alert service → location, siren, and SMS compose action. An SOS button can independently enter the same alert path.

D. *Scope of implemented features*

The introduction and system design chapters include uncommon wake-word activation, Graph Neural Network (GNN) risk prediction, Firebase monitoring, background operation, and direct notifications to emergency authorities. In contrast, Chapter 5 describes wake-word support and background/screen-off operation as deferred development work; it does not establish a trained or deployed GNN. These proposed design features are excluded from the implemented-system claims in this article.

E. *Runtime and software environment*

The source identifies Flutter and Dart for application development, record for microphone capture, permission_handler for runtime permissions, shared_preferences for on-device contact storage, geolocator for location lookup, audioplayers for siren playback, url_launcher for SMS and call intents, and tflite_flutter for local inference. Python, TensorFlow/Keras, and

Google Colab are identified for model development. The exact package versions, host GPU, device model, installed memory, epochs, batch size, optimizer, and learning rate are not established in the provided report; asserting values would prevent faithful reproduction.

V. AUDIO MODEL AND MUFFLING METHODOLOGY

The stated model design classifies input audio into normal and distress-related categories. Its intended front end computes a mel-spectrogram from the waveform, followed by CNN processing and a sigmoid output. The model contract places preprocessing within the model to avoid duplicating it in Dart. The report identifies RAVDESS categories fearful and angry as positive, with neutral, calm, and happy as negative. Dataset partition sizes, actor-level separation, optimization settings, and final model architecture are not documented comprehensively.

For muffling augmentation, the reported experimental plan applies low-pass filtering, additive noise, and gain reduction to simulate acoustically obstructed recordings. The intended controlled comparison trains one baseline model without these transformations and one augmented model with them, then evaluates both on clean and muffled held-out audio. Reproducible verification requires fixed train/validation/test membership, actor-disjoint splits, augmentation parameters, a recorded random seed, consistent thresholds, and confusion matrices.

The augmentation is designed to approximate plausible acoustic changes when a phone is partially covered. Low-pass filtering reduces the contribution of higher-frequency speech information, attenuated gain lowers signal amplitude, and additional noise lowers the signal-to-noise ratio. Applied too strongly, however, these manipulations can create an artificial distribution that differs from recordings captured through actual fabrics or bags. Parameters such as filter cutoff, noise level, probability of applying each transform, and gain range must be reported before the technique can be replicated.

The most informative comparison fixes the original clean-recording partition before augmentation. Recordings from the same actor or original performance should not appear in both training and evaluation through separate transformed copies.

Otherwise the model may recognize speaker-specific or clip-specific information, exaggerating generalization. If the study uses RAVDESS actor labels, an actor-disjoint partition is a defensible evaluation choice, though the original report does not specify that such a split was actually used.

For distress classification, recall = TP/(TP + FN). Improvement in recall is the augmented-

model recall minus baseline recall, evaluated on the identical withheld test set. For actual deployment, recall alone is insufficient: false alarms per hour, precision, inference latency, battery cost, device diversity, and audio privacy must also be measured.

A. Experimental evaluation procedure

For a reproducible comparison, the same actor-disjoint held-out clips should be used for both baseline and augmented classifiers under matched preprocessing, class assignments, and threshold selection. Each run should record dataset membership, random seed, augmentation filter cutoff and gain/noise parameters, final network dimensions, training schedule, and model output tensors. Predictions on clean and artificially muffled conditions should be saved to compute confusion matrices, precision, recall, F1, accuracy, and the false-positive rate. This is a documented evaluation requirement, not a claim that all these measurements have been completed.

Each model should be assessed on the same unaugmented and muffled held-out clips, using identical label definitions and predetermined threshold rules. A confusion matrix supplies true positives, false negatives, false positives, and true negatives; from these, distress recall, precision, F1-score, and specificity can be calculated. The number of evaluated clips and actors, along with uncertainty intervals or repeated runs, should accompany point estimates. A baseline-to-augmented change is only attributable to augmentation if other substantive choices, including training duration and architecture, are held comparable.

Testing should also isolate the post-classifier decision logic. Consecutive-hit filtering may reduce one-window false alarms but can delay an alert or miss short vocal events. Its behavior is best quantified in events rather than individual windows: event-level sensitivity, alerts per hour of ordinary ambient sound, and elapsed time from event onset to countdown initiation. No event-level benchmark or time measurements are supplied in the source report.

B. Measurement protocol and sources of error

The report proposes evaluation on a held-out test set using recall, precision, F1-score and a confusion matrix. Because distress labels are inferred from acted emotions in RAVDESS, subject identity and recording conditions can confound a random file-level split. Any final comparison should document actor-disjoint train/validation/test allocation, class counts, augmentation applied only to training samples, and matched test conditions. A smartphone-based trial should also measure processing latency, background noise sensitivity, battery consumption, false alerts per hour and SMS handoff success.

These are evaluation requirements, not measured outcomes in the supplied report.

C. Model-to-application contract

The documented model interface expects mono 16-kHz normalized floating-point audio and a single distress score. Inference runs via tflite_flutter behind an interchangeable detector interface. The report specifies incompatible waveform lengths of 48,000 and 32,000 samples, and different threshold values; therefore neither is presented as a confirmed final training or deployment parameter. These settings must be checked against the actual exported TensorFlow Lite signature and Dart listening service before the experiments can be reproduced.

VI. RESULTS AND DISCUSSION

The uploaded report provides a 25-case functional and integration testing table in which each listed case has status "Pass." These cases concern application workflows and expected behaviors such as contact handling, permissions, SOS flow, location, siren, and audio/alert integration. Accordingly, the report records a 25/25 or 100% pass fraction for its listed test scenarios. This fraction is not a machine-learning accuracy score, not proof of robust field performance, and was not independently reproduced for this manuscript.

The functional test list principally establishes documented expected-versus-observed behavior for the prototype's interfaces. It does not measure the probability that distress in the environment is detected correctly, nor does it demonstrate that an SMS reached an emergency contact. A 25/25 pass entry should therefore be understood as the project team's reported status for listed test cases, not as an independently reproduced reliability percentage for the deployed safety system.

For a reviewer, the most useful functional evidence would be a traceable test matrix linking each requirement to an action, expected response, observed response, and build identifier.

Representative cases include adding and deleting a trusted contact, starting and stopping microphone capture, denying a location permission, using manual SOS while the classifier is unavailable, and canceling a countdown before a messaging handoff. The uploaded report lists functional cases and outcomes but does not supply an automated execution trace or independently replayable evidence for all of them. Therefore the discussion distinguishes reported test status from observed measurements.

The separation between unit-level behavior and operational success is especially important for communication functionality. Constructing a location-bearing SMS body, opening a device messaging application, sending a message over a cellular network, and verifying the recipient received it are four distinct steps with distinct failure modes. Offline service, denied

permissions, invalid phone numbers, or a user abandoning the composer can interrupt the process. A future validation study should record completion status at each step rather than summarize the complete workflow with a single pass indicator.

The reported difference between 89.6% and 72.4% corresponds to 17.2 percentage points. This is a descriptive arithmetic comparison only. Without the sample denominator, confusion matrices, test-set composition, and run-level outputs, the precision of the estimates and their statistical significance cannot be judged. The result also cannot establish that the improvement would hold for whispered sounds, naturally occurring threats, different microphones, or unseen demographic groups.

The functional and model-related observations are distinct: (i) 25 of 25 functional/integration cases are marked as passed in the report; (ii) the baseline model is listed at 72.4% muffled-distress recall; and (iii) the augmented model is listed at 89.6%. Their reported recall difference is 17.2 percentage points. The latter two values remain unverified because the primary experiment records are not supplied.

A model comparison table in Chapter 5 reports 72.4% recall on muffled distress for the baseline and 89.6% for the augmented model. Yet the adjacent prose explicitly states that dataset split, augmentation values, training configuration, and experimental results remain to be filled. Without supporting predictions or a reproducible evaluation record, the percentages cannot be independently verified. They are retained here only to characterize the source document, not as a substantiated effectiveness claim.

TABLE I. SOURCE-REPORTED OBSERVATIONS (UNVERIFIED)

Source-reported item	Value	Evidence status
Functional / integration tests	25/25 pass	Reported; not rerun
Baseline muffled-distress recall	72.4%	Raw predictions absent
Augmented muffled-distress recall	89.6%	Raw predictions absent
Recall gain	17.2 pp	Calculated from reported values

VII. LIMITATIONS AND PUBLICATION REQUIREMENTS

First, emotional speech is an imperfect proxy for dangerous real-life situations, and a safety classifier should not be represented as capable of recognizing all emergencies. Second, the audio window length and operating threshold are inconsistent across report sections. Third, the model weights, training configuration, confusion matrices, and held-out testing logs are not part of the supplied material. Fourth, the SMS action is user-confirmed and cannot be described as guaranteed autonomous notification. Fifth, in-app listening is documented, but unattended background monitoring, uncommon wake-word recognition, and the GNN

component are not established as implemented.

Privacy and consent form additional deployment constraints. The planned use of device-side inference reduces the need to transmit raw audio, but the report does not provide a formal threat model, retention audit, or consent study for potentially sensitive nearby conversations. A defensible production design should make listening state prominent, avoid unneeded recording persistence, limit access to saved contacts and locations, and explain to users that acoustic predictions are fallible.

The current documentation also lacks a cross-device compatibility matrix. Android manufacturers and operating-system versions can differ in microphone capture permissions, location-provider behavior, battery restrictions, and the availability of URI handlers for calls and messaging. A repeatable evaluation should therefore name the device models, Android versions, recording hardware, and software build used for each result. Without these conditions, even a successful demonstration is difficult to generalize to another user's handset.

Risk communication is part of the user interface design. The screen should explain whether listening is active, whether an alert countdown is underway, whether location data is available, and whether the outgoing communication still requires user approval. These separate states help prevent confusion between an attempted alert and a completed notification. The project report documents aspects of the interface, but it does not include a controlled usability study under simulated stress or an accessibility assessment of the notification interactions.

A clinical or public-safety interpretation would also require broader testing than the supplied material supports. A false alarm may expose personal location or create unnecessary concern, whereas a missed alarm can delay help. Threshold calibration therefore calls for user-centered assessment, transparent override and cancellation controls, and testing in noisy, quiet, indoor, and outdoor conditions. These items are recommendations for future evaluation, not outcomes reported by the project.

Before submission to a peer-reviewed journal, authors should reconcile model input and threshold settings against the actual repository, provide the experimental protocol and raw confusion-matrix counts, document consent and privacy safeguards for audio capture, test across real phones and acoustic conditions, confirm author names and affiliations, and review the final target journal's author instructions.

VIII. CONCLUSION AND FUTURE WORK

Abhaya demonstrates an implementation pattern for integrating a local audio detector into an Android-first personal safety application without tightly coupling model development to emergency-response code. The report supports a modular Flutter application, a TensorFlow Lite detector interface, a shared SOS workflow, and a stub-based test process. Its proposed muffling-augmentation comparison is technically relevant to concealed-device operation, but robust accuracy claims await reproducible model and field evaluation. Future work should validate the actual model contract, strengthen testing with speaker-disjoint and device-based trials, and assess background monitoring and alert-delivery guarantees before any safety-critical deployment.

The recommended next engineering milestone is to freeze and publish one consistent detector contract, verify the exported model tensor dimensions, and run repeated on-device inference tests on a specified handset. The corresponding research milestone is an actor-disjoint, predeclared evaluation using the identical held-out clips for the baseline and muffling-aware models. Sharing the confusion matrices, label map, augmentation configuration, training script, and raw prediction outputs would make the model comparison auditable.

Additional testing should introduce actual pocket and bag recordings, ambient sound mixtures, different speaker characteristics, and a clearly distinguished wake-word subsystem if that feature is ultimately implemented. Device-level evaluations should measure battery impact, model loading time, inference latency, false alerts per monitoring hour, and time from a simulated distress cue to the user-visible alert.

These are necessary steps toward evidence-backed deployment, not claims that the current prototype has already met such requirements.

REFERENCES

1. S. R. Livingstone and F. A. Russo, “The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English,” PLOS ONE, vol. 13, no. 5, e0196391, 2018, doi: 10.1371/journal.pone.0196391.
2. Flutter, “Flutter documentation,” https://docs.flutter.dev/ (accessed Oct. 23, 2026).
3. Picovoice, “Porcupine wake word: Flutter quick start,” https://picovoice.ai/docs/quick-start/porcupine-flutter/ (accessed Oct. 23, 2026).
4. Abhaya project team, “Abhaya—AI-Driven Women Safety System using Wake Word Detection and Muffled Sound Recognition,” unpublished major-project report, Department of Computer Science and Engineering, JIT, 2026–2027.