



Audio Visual Emotion Detection System

Katikidala Gowthami Sri, Dr. K. Jhansi Rani¹

¹ M.Tech., Scholar, Associate Professor

Department of Electronics and Communication Engineering, University College of Engineering Kakinada(A), JNTUK, Kakinada, Andhra Pradesh, India

ABSTRACT

Emotions unfold as continuous streams rather than isolated snapshots, and research into detecting the temporal boundaries of emotion segments in audio-visual signals has produced multi-stage pipelines involving boundary proposal, classification, completeness checking, and overlap suppression ([Li et al., 2024](#)). Reproducing such pipelines end to end typically requires a trained multimodal model and a curated dataset, which may place hands-on exploration of the workflow beyond the practical reach of most learners and early-stage developers. This report documents AVES Pulse, a native Android application built with Kotlin and Jetpack Compose that reconstructs the shape of this workflow as an interactive, fully local mobile tool. The application enables users to capture or select a video, observe its progression through a five-stage simulated detection pipeline, and inspect the resulting emotion segments on an interactive timeline, run multi-annotator agreement analysis using Fleiss' Kappa, and benchmark configurations using temporal Intersection-over-Union (tIoU) and Average Precision (AP) / mean Average Precision (mAP). The application follows a Model-View-ViewModel (MVVM) architecture with Room for local persistence, CameraX and AudioRecord for live audio-visual capture, and an optional cloud language-model call as one data source among several. Verification against the app's own bundled demo data suggests that the statistical modules — Fleiss' Kappa, tIoU, and AP/mAP — appear to be computed correctly, while the detection stages themselves appear to remain a UI-level simulation intended for demonstration, teaching, and rapid prototyping rather than to constitute a trained model. The report describes the system architecture, the pipeline design, the statistics implemented, and a roadmap toward replacing the simulated stages with genuine on-device inference.

Keywords: *Android, Jetpack Compose, MVVM Architecture, Emotion Stream, Fleiss' Kappa, Temporal IoU, Average Precision, Room Database, Mobile Prototyping*

I. INTRODUCTION

Human emotional expression rarely arrives as a single, isolated label — it may change shape over the course of a conversation, a reaction video, or a recorded interaction. Understanding where one emotional state ends and another begins, rather than simply classifying a pre-trimmed clip, may be understood as the central premise of temporal emotion-stream detection research. Reproducing a research-grade pipeline for this task, however, generally depends on a trained multimodal encoder, a curated annotated dataset, and a graphics processing unit (GPU)-backed inference environment — resources that are not readily available to a student or an independent developer who wants to understand how such a system is structured.

AVES Pulse was designed to address that gap at the interaction-design level. Rather than training a model, the application reconstructs the shape of a boundary-based temporal detection workflow as a fully operational Android application: a user supplies a video (from the gallery, a live camera and microphone recording, or a bundled demo clip), the app steps through five clearly named pipeline stages, and the output is rendered as an interactive segment timeline that can be annotated, cross-checked for inter-rater agreement, and benchmarked. The goal is threefold: (1) to give a concrete, touchable mental model of what a boundary-proposal-based detection pipeline looks like from the outside, (2) to implement the surrounding statistical machinery — inter-annotator agreement and detection-quality metrics — with genuine, verifiable correctness, and (3) to provide a working scaffold that a future iteration can extend with real on-device inference.

II. RELATED WORK

The application's five-stage pipeline naming (encode, propose boundaries, classify, verify completeness, suppress overlaps) appears to be conceptually modelled on boundary-based temporal action/emotion detection research ([Yang et al., 2023](#)), which frames detection as proposal generation followed by classification and redundancy removal ([Liu et al., 2022](#)). Two building blocks used inside the app appear to derive from established statistical methods: Fleiss' Kappa for measuring agreement among more than two raters ([Hallgren, 2012](#)), and Soft Non-Maximum Suppression for score-decaying overlapping detections rather than deleting them outright ([Bodla et al., 2017](#)). The app's evaluation module also appears to follow the standard temporal Intersection-over-Union and Average Precision / Mean Average Precision formulation used broadly in temporal detection benchmarks ([Alwassel et al., 2018](#)). These are used here as design references for the app's structure and terminology, not as claims that a trained detection network is present in the current build.

III. METHODOLOGY

3.1 System Architecture

The application appears to implement the MVVM (Model-View-ViewModel) pattern. Compose screens observe state exposed by a single AppViewModel, which appears to delegate responsibility for all data operations to an AppRepository. The repository may function as the principal source of truth, providing access to two backends: a local Room database (media assets, emotion segments, annotation sessions, agreement scores, and evaluation runs) and, optionally, a remote network call used only as one possible way to source segment data. Device sensors — CameraX for the live preview and AudioRecord for microphone amplitude sampling — are accessed directly from the relevant composables; their outputs are subsequently propagated to the same ViewModel. A dedicated math utility object implements the three statistical routines (Fleiss' Kappa ([Fanaee-T & Gama, 2013](#)), temporal IoU, Average Precision (mAP) ([Hirling et al., 2023](#))) used throughout the annotation and benchmarking screens.

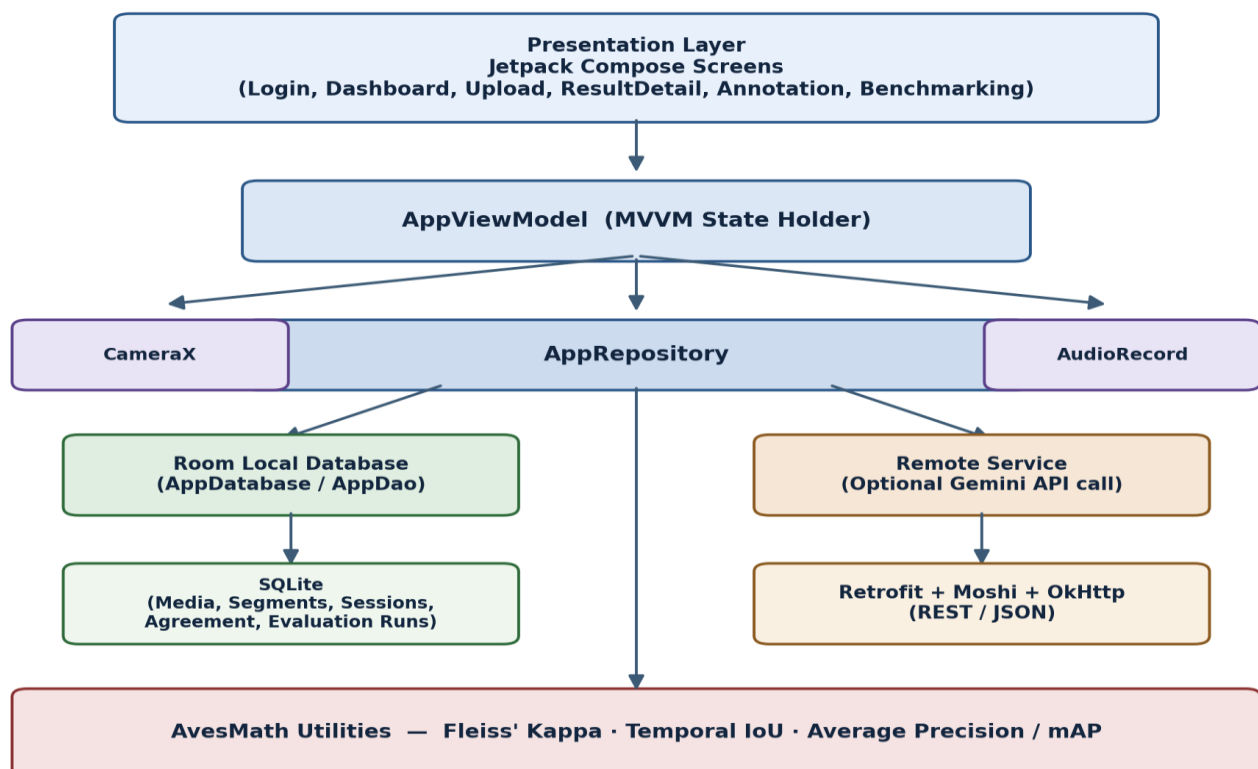


Fig. 1. System architecture of AVES Pulse (MVVM).

3.2 Technology Stack

Layer	Technology	Purpose
User interface (UI)	Jetpack Compose and Material 3	Declarative screen composition and theming
Navigation	Compose Navigation	Routing among login, dashboard, upload, detail, and benchmark screens
State management	AndroidViewModel + StateFlow	Single ViewModel driving all screens
Persistence	Room (SQLite)	Local storage of media, segments, sessions, scores, runs

Networking (optional)	Retrofit + Moshi + OkHttp	Optional call to a generative language model
Camera capture	CameraX	Live camera preview during recording
Audio capture	AudioRecord (PCM)	Real-time microphone amplitude sampling
Statistics	Custom Kotlin utilities	Fleiss' Kappa, temporal IoU, AP / mAP

3.3 Simulated Detection Pipeline

Upon submission of a video, the application progresses sequentially through five delineated stages, each rendered to the user in the form of a progress row, accompanied by a real-time status indicator. The stages are, in order: encoding the visual and audio streams, proposing temporal boundaries, running the emotion classification stage, checking completeness of each proposed segment, and pruning overlapping proposals after which the final segment list is committed to the database. This staged presentation appears intended to render the architecture of a boundary-based detection pipeline legible to the user, irrespective of the underlying computational mechanisms.

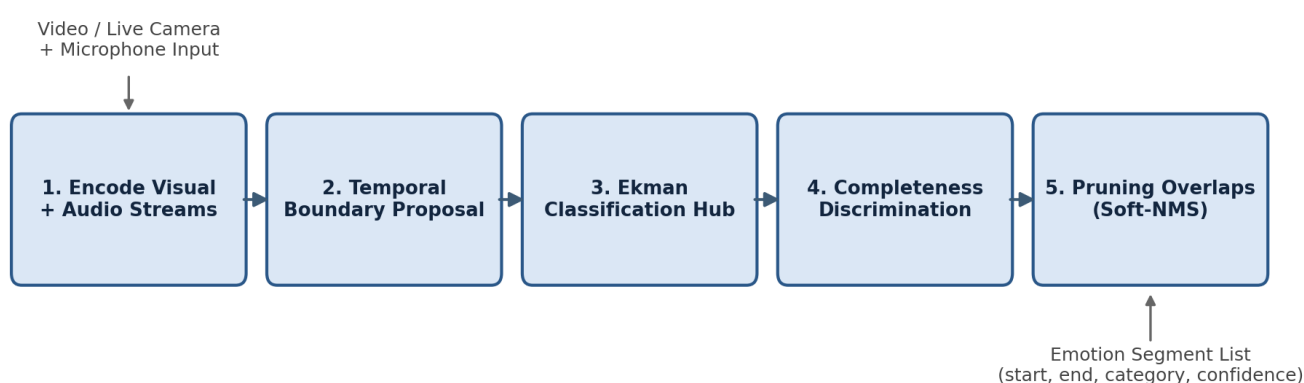


Fig. 2. The five-stage simulated detection pipeline as presented to the user.

3.4 Data Model

Entity	Key Fields	Role
MediaAsset	id, title, durationS, fps, status, uri	A single uploaded, recorded, or demonstration video
EmotionSegment	mediaId, source, startS, endS, category, confidence	One labeled time span; source distinguishes predicted / ground_truth / annotator_1 / annotator_2
AnnotationSession	mediaId, annotators	Groups the raters assigned to a single video
AgreementScore	sessionId, kappa	Stored Fleiss' Kappa result for a session
EvaluationRun	config, mAP, wAP, timestamp	Results from a single benchmarking run

3.5 Statistical Evaluation Modules

Three statistical routines appear to be implemented directly in Kotlin and reused across the Annotation and Benchmarking screens.

Fleiss' Kappa may be used to quantify agreement among more than two raters across a fixed set of categories (Moons & Vandervieren, 2025). For N items rated by n raters into k categories, with n_{ij} the number of raters assigning item i to category j :

$$\kappa = (\bar{P} - \bar{P}_e) / (1 - \bar{P}_e)$$

Here, $\bar{P}$ denotes the mean observed per-item agreement, whereas $\bar{P}_e$ denotes the agreement expected under chance, as computed from the marginal category proportions $p_{.j}$ across all $N \times n$ ratings.

Temporal Intersection-over-Union (tIoU) may be used to quantify the extent to which a predicted segment $[s_1, e_1]$ overlaps a ground-truth segment $[s_2, e_2]$ (Vahdani & Tian, 2022):

$$tIoU = |\text{overlap}(s_1, e_1, s_2, e_2)| / |\text{union}(s_1, e_1, s_2, e_2)|$$

Average Precision (AP) and mean Average Precision (mAP) are conventionally calculated through the standard confidence-ranked precision-recall formulation used in temporal detection benchmarks (Alwassel et

al., 2018): predictions are sorted by confidence, matched greedily to unmatched ground-truth segments of the same category whose temporal Intersection over Union (tIoU) exceeds a chosen threshold, and the area under the resulting precision-recall curve is taken as the AP for that category and threshold; mAP may be understood as the average of AP across categories.

IV. APPLICATION WALKTHROUGH AND VERIFICATION

4.1 Screens Overview

Screen	Function
Login	Local mock authentication (via a profile picker or a custom name/email)
Dashboard	Provides a searchable inventory of processed media assets and summary statistics
Upload	Source selection (gallery, live capture, or demo template) and pipeline-progress visualisation
Result Detail	Segment timeline, duration histogram, and JavaScript Object Notation (JSON) export
Annotation Studio	Multi-annotator segment comparison and live Fleiss' Kappa
Benchmarking	Executes and stores evaluation configurations with mean and weighted Average Precision (mAP and wAP) scores

4.2 Demo Content and Timeline Visualization

The application includes four bundled reaction-style templates (a movie-trailer reaction, a comedy-show reaction, a horror-game reaction, and a durian-tasting vlog), thereby facilitating exploration of the pipeline and downstream screens without external media. The segment categories generated for these four templates are shown in Fig. 3; happiness and surprise appear to dominate, which may reflect the reaction-video content represented by the templates.

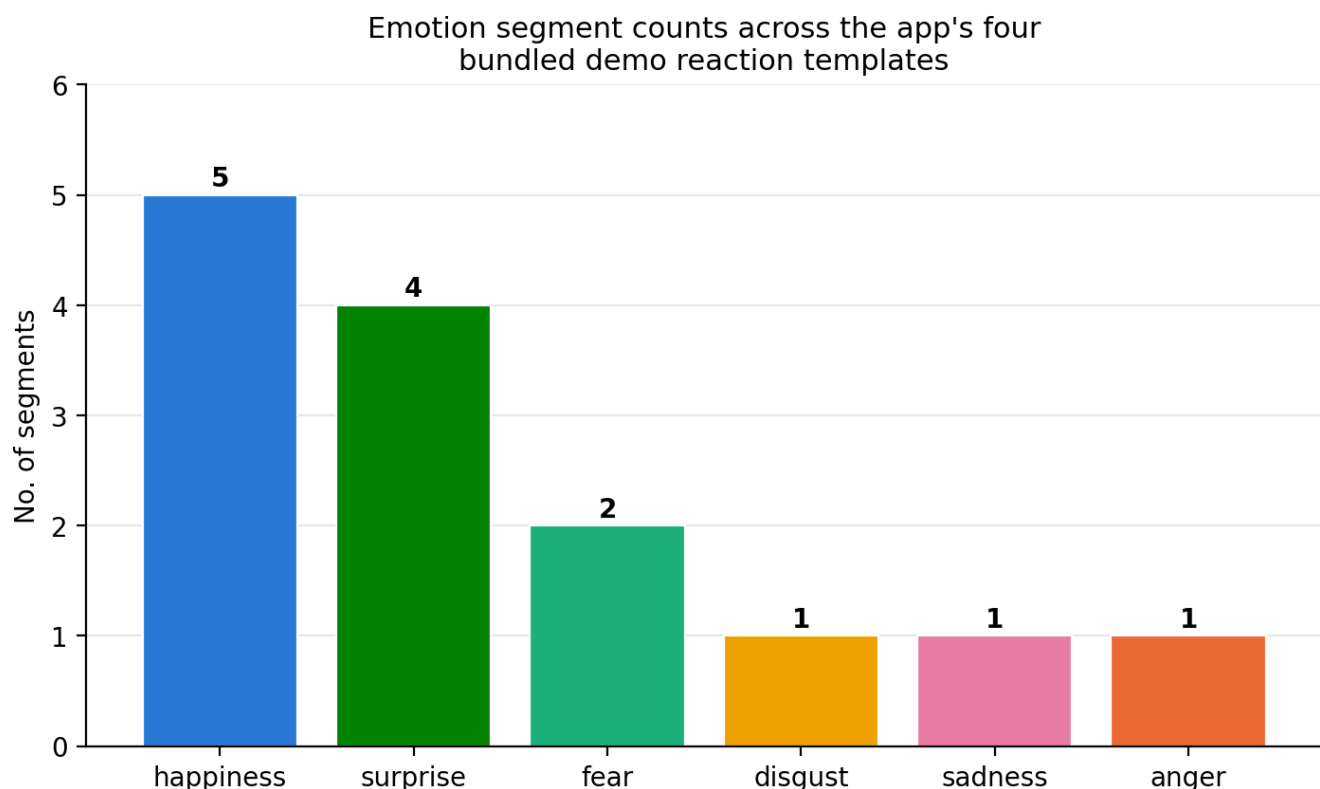


Fig. 3. Emotion segment counts across the app's four bundled demo templates.

Fig. 4 delineates the complete segment timeline associated with the default “Durian Tasting Vlog” demo asset: the ground-truth track, the pipeline's predicted track, and two independent synthetic annotator tracks, as stored directly in the app's Room database.

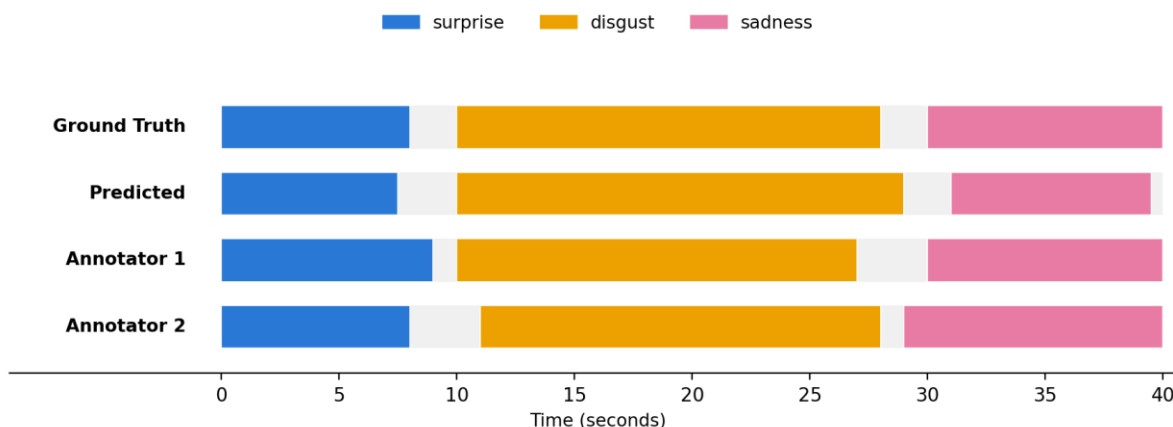


Fig. 4. Ground truth, predicted, and two annotator tracks for the bundled Durian demo asset.

4.3 Verification of Statistical Modules

To assess whether the app's statistics correspond to real bundled data (rather than illustrative numbers), the Fleiss' kappa ([Kılıç, 2015](#)) and temporal intersection-over-union (tIoU) ([Mikkelsen et al., 2016](#)) routines were evaluated directly against the Durian demo asset's stored segments.

Metric	Input	Result
Fleiss' Kappa	Ground truth + Annotator 1 + Annotator 2 tracks (3 raters, 4 categories, 40 one-second bins)	$\kappa = 0.904$ (Almost Perfect Agreement (Zapf et al., 2016))

Category	Predicted Span	Ground-Truth Span	temporal Intersection over Union (tIoU)	Localized at tIoU ≥ 0.5
Surprise	0.0s – 7.5s	0.0s – 8.0s	0.94	Yes
Disgust	10.0s – 29.0s	10.0s – 28.0s	0.95	Yes
Sadness	31.0s – 39.5s	30.0s – 40.0s	0.85	Yes

All three predicted segments appear to exceed the standard 0.5 tIoU threshold ([Zou et al., 2023](#)) relative to their corresponding ground-truth spans, and the resulting Fleiss's Kappa falls in the “almost perfect agreement” band. These results suggest that the statistical layer of the app — as opposed to the simulated detection stages themselves — produces mathematically valid output on the app's own data.

V. CONCLUSION

AVES Pulse encapsulates the structural form of a boundary-based temporal emotion-stream detection workflow into a self-contained, fully local Android application. Its MVVM architecture, Room-backed persistence, and live camera/microphone capture appear to be functionally operational, and its statistical core — Fleiss' Kappa for inter-annotator agreement and temporal IoU / AP / mAP for detection evaluation — appears to compute correctly on the app's own bundled data. The detection pipeline itself remains a five-stage simulation rather than a trained model, and the report is explicit about that boundary throughout. As a teaching and prototyping tool, the application appears to make an otherwise abstract multi-stage detection workflow concrete, inspectable, and interactive on a phone.

VI. FUTURE SCOPE

Future versions of AVES Pulse may integrate real multimodal feature extraction and physiological sensing.

Figure 6.1 Proposed Feature Extraction Pipeline

Input video and audio -> Visual & Audio Features -> Temporal Boundary Detection -> Emotion Classification -> Completeness Verification -> Soft-NMS -> Future Physiological Features -> Final Emotion Timeline

Table 6.1 Visual Feature Extraction

Feature	Technique	Purpose
Facial Action Units	AU1, AU4, AU12, AU15	Facial muscle movements
Face++	Landmarks	Face geometry
TSAV	Two-Stream Encoder	Visual embeddings
Data2Vec	Self-supervised	Deep visual features

Table 6.2 Audio Feature Extraction

Feature	Technique	Purpose
Mel-frequency cepstral coefficients (MFCC)	Cepstral Coefficients	Speech and pitch
TSAV Audio	Audio encoding	Acoustic Embeddings

Table 6.3 Temporal Detection

Stage	Output	Purpose
Boundary Proposal	Start/end timestamps	Detection of emotion intervals
Classification	Seven emotions	Emotion recognition
Completeness	Validated Segments	Removal of incomplete detections
Soft-NMS	Filtered Output	Duplicate suppression

Table 6.4 Future Modalities

Feature	Measures	Application
Remote photoplethysmography (rPPG)	Blood volume pulse / beats per minute (BVP/BPM)	Heart-rate estimation (Chin et al., 2026)
Thermal mapping	Facial temperature	Stress detection (Gioia et al., 2022)
Vocal micro-tremor	8–14 Hz	Hidden stress (Amirian, 2023)
Skeleton Tracking	Posture and fatigue	Behavior analysis (Alfavo-Viquez et al., 2023)

REFERENCE:

- [1] Beyond these modalities, the integration of label distribution learning could offer a more nuanced approach to continuous emotion prediction compared to discrete classification ([Schoneveld et al., 2021](#)).
- [2] Integrating these physiological modalities could enable the development of a Bayesian framework for modeling uncertainty in emotion state predictions, allowing users to calibrate acceptance boundaries based on confidence thresholds ([Harper & Southern, 2020](#)).
- [3] Android Developers, “Jetpack Compose documentation,” [developer.android.com/jetpack/compose](#). Furthermore, incorporating inter-annotator reliability metrics into the evaluation process would facilitate the detection of outliers and improve overall model performance ([Baveye et al., 2015](#)).
- [4] Android Developers, “Room persistence library documentation,” [developer.android.com/training/data-storage/room](#).
- [5] Android Developers, “CameraX documentation,” [developer.android.com/training/camerax](#).