Natural Resources for Human Health
eISSN: 2583-1194

Original Research

Received 24 May 2026
Revised 28 June 2026
Accepted 16 July 2026

Natural Resources for Human Health 2026, Vol. 6 (7s): 867–879

KEYWORDS:
YOLOv8, Alcohol Detection, Transfer Learning, EasyOCR, Real-Time Object Detection, OpenCV, Socket.IO, Brand Identification, Precision-Recall Analysis.

AI-Powered Real-Time Alcohol Detection Using Fine-Tuned YOLOv8 and OCR-Based Brand Identification

Anilkumar BH[1], Puttegowda D[2]

[1]Assistant Professor, Department of Computer Science and Engineering, Vidyavardhaka College of Engineering, Mysuru, India. Email: anilkumar.bh@vvce.ac.in
[2]Department of Computer Science and Engineering, ATME College of Engineering, India. Email: puttegowda.77@gmail.com
* Corresponding Author Email: anilkumar.bh@vvce.ac.in

ABSTRACT: This paper presents an AI-powered real-time alcohol detection system built around a YOLOv8n object detector fine-tuned via transfer learning from COCO-pretrained weights, combined with OpenCV-based frame acquisition and an OCR-assisted brand identification layer. The detector was fine-tuned on a public 857-image alcohol/no-alcohol dataset (Roboflow Universe) for 40 epochs on a GPU (NVIDIA Tesla T4, Google Colab), following an initial 15-epoch CPU run that was found to be undertrained. On a genuine held-out test set of 87 images (137 independently verifiable ground-truth boxes), the fully converged fine-tuned model achieves a mAP@0.5 of 0.829 and an F1-score of up to 0.831, compared to 0.364 mAP@0.5 and 0.508 F1 for a stock COCO-pretrained baseline restricted to generic "bottle", "wine glass", and "cup" classes — a 128% relative improvement in mAP@0.5 from fine-tuning alone. Three additional components operate on top of detection: an optional EasyOCR label reader coupled with fuzzy keyword matching against a brand database covering major global and Indian alcohol brands; a 180°-rotation fallback inference pass that recovers detections of knocked-over or inverted containers; and a consecutive-frame debounce filter that suppresses single-frame false triggers. Detections are relayed over Socket.IO to a React dashboard and an OS-level popup alert. Deployment inference runs at approximately 22–24 FPS on standard hardware. We report full training convergence curves, a threshold-swept precision–recall analysis, an explicit detection confusion breakdown, and the system's genuine limitations.

1. INTRODUCTION

The rapid proliferation of online media and live video streaming has expanded viewer exposure to alcohol imagery, particularly among youth populations. Research consistently links exposure to alcohol-related visual content with increased alcohol consumption and risk behaviour. Manual moderation cannot scale to match the volume of user-generated live streams; automated, scalable alternatives are needed.

Existing general-purpose content moderation systems — such as Amazon Rekognition and Google Cloud Vision AI — provide broad object labelling, nudity, or violence detection pipelines, but none incorporates a dedicated, purpose-built alcohol-detection module. This paper addresses that gap with a modular, real-time alcohol detection system built on the YOLOv8 object detection framework, OpenCV frame processing, and an OCR-assisted brand identification layer, and reports a fully transparent, reproducible evaluation of that system rather than projected or assumed performance.

The core contributions of this work are fivefold:

- Fine-tuning YOLOv8n via transfer learning on a public alcohol/no-alcohol dataset, taken to full training convergence (40 GPU epochs, verified against the loss and mAP curves) after an initial under-trained 15-epoch run was identified and corrected.
- An OCR-plus-fuzzy-keyword brand identification algorithm (Algorithm 2) that generalizes to brands — including Indian brands — never present as trained detection classes.
- A 180°-rotation fallback inference pass that recovers detections of inverted or knocked-over containers missed on the first pass.
- A full quantitative evaluation: baseline-vs-fine-tuned comparison, a nine-point confidence-threshold sweep with a precision–recall curve, and an explicit true/false-positive/negative detection breakdown, all computed on a genuine held-out test set rather than assumed.
- A transparent account of the system's real limitations and of a data-quality issue discovered during evaluation (mixed-format ground-truth labels), rather than presenting only favourable results.

2. PRELIMINARIES

2.1 YOLOv8 Object Detection Framework

YOLO (You Only Look Once) comprises a family of single-stage, real-time object detectors that reformulate image detection as a single regression problem over a grid of bounding boxes and class probabilities. YOLOv8 introduces an anchor-free detection head with decoupled classification and regression branches, a CSPNet-derived backbone (C2f module) with improved gradient flow, and a Feature Pyramid Network with Path Aggregation Network (FPN+PAN) neck for multi-scale feature fusion. The nano variant (YOLOv8n) used in this work has 3,011,238 parameters and requires 8.2 GFLOPs at 640×640 resolution, making it suitable for both cloud training and modest inference hardware.

2.2 Transfer Learning

Transfer learning enables higher detection accuracy without requiring millions of training samples, by initialising model weights from a pre-trained checkpoint — typically one trained on COCO — and fine-tuning on a smaller, domain-specific dataset. This system initialises from COCO-pretrained YOLOv8n weights (of 355 total parameter tensors, 319 were successfully transferred) and fine-tunes on an 857-image alcohol-specific dataset. Section 7.2 shows this was sufficient to reach a plateaued, converged state within 40 epochs.

2.3 OpenCV Frame Processing and Pre-Checking

OpenCV handles frame acquisition (webcam, video file, or live screen capture) and a lightweight pre-check stage prior to inference: each frame is converted to grayscale to compute mean luminance, and near-black or heavily occluded frames (mean luminance below 15) are skipped rather than passed to the detector. A frame-differencing motion score is also computed per frame for downstream diagnostics.

2.4 EasyOCR and Fuzzy Text Matching

Class-name-only brand matching cannot distinguish between brands, since a generic detector only outputs labels such as "bottle". To address this, an optional EasyOCR reader extracts label text from the cropped detection region. The extracted text is matched against a static brand keyword list using both direct substring matching and fuzzy word-level matching (Python difflib SequenceMatcher, ratio ≥ 0.7), which tolerates OCR misreads without requiring the brand to have been a trained detection class. This procedure is formalised in Algorithm 2.

2.5 Debounce-Based False-Positive Suppression

Relying on a single-frame detection decision is prone to transient false triggers. This system applies a simple consecutive-frame debounce: an alert fires only once a positive detection has occurred on three consecutive frames, and the counter resets immediately on any frame without a detection. This is a simpler alternative to windowed-ratio smoothing, trading some robustness to intermittent flicker for implementation simplicity and low latency.

Natural Resources for Human Health
eISSN: 2583-1194

3. PROPOSED METHODOLOGY

3.1 System Overview

The proposed system follows a five-stage pipeline: (1) frame acquisition, (2) luminance/motion pre-check, (3) two-pass YOLOv8n detection with a 180°-rotation fallback, (4) OCR-assisted brand identification and a person-overlap drinking-action heuristic, and (5) consecutive-frame debounce followed by alert dispatch. Each stage is implemented as an independent, swappable module (Section 6.2).

3.2 Dataset

The model was fine-tuned on the public "Alcohol Detection" dataset from Roboflow Universe [12], comprising 857 annotated images across two classes: alcohol and no-alcohol. The dataset was split 80/10/10 into 685 training, 85 validation, and 87 test images. During training, 20 training images and 1 validation image were automatically excluded by Ultralytics because their label files mixed segmentation-polygon rows with detection-box rows — an artefact of the source export rather than a pipeline error. The same issue affects 7 of 144 ground-truth alcohol boxes in the test set (Section 7.1), which the evaluation script correctly excludes from its 137-box denominator rather than silently mis-parsing them. Standard Ultralytics training-time augmentations were applied, including mosaic composition, HSV colour jitter, horizontal flip, translation, and scaling.

3.3 Model Training

Training used the YOLOv8n (nano) variant, initialised from COCO-pretrained weights. An initial fine-tuning run of 15 epochs on CPU-only hardware was performed first; inspection of its loss and validation-mAP curves showed both were still improving with no plateau, indicating the run was stopped before convergence rather than because it had converged. Training was therefore repeated for 40 epochs on a GPU (NVIDIA Tesla T4, Google Colab), completing in 0.124 hours (≈ 7.4 minutes) — a large practical speed-up over the CPU run that made the longer schedule feasible. Batch size was 16 at 640×640 input resolution. Ultralytics' automatic optimizer selection ('optimizer=auto') chose AdamW with an initial learning rate of 0.001667 and momentum 0.9, overriding the manually specified SGD-style settings (lr0=0.01, momentum=0.937) used in the earlier CPU run; this is expected behaviour of the auto-selection logic rather than a configuration error, and is reported here for reproducibility rather than treated as a controlled hyperparameter.

3.4 Inference and Alert Pipeline

At deployment inference, each frame is passed through the fine-tuned model at a confidence threshold of 0.20, with non-maximum suppression handled internally by Ultralytics (IoU = 0.7). If no bottle/alcohol class is found in the upright pass, the frame is rotated 180° and re-inferred, with detected coordinates remapped back to the original orientation. Each detection above threshold is cropped and passed to Algorithm 2 for brand identification and drinking-action flagging. Detections satisfying the 3-frame debounce (Algorithm 1) trigger a Socket.IO alert, an OS-level popup with sound, and an event log entry.

4. PROPOSED ARCHITECTURE

LAYER 1: INPUT LAYER
Video Source Acquisition
Webcam Feed | Uploaded Video File (MP4/AVI) | Live Screen Capture (mss)
▼ Raw Frame (FRAME_SKIP = 1)
LAYER 2: PRE-CHECK LAYER (OpenCV)
Luminance & Motion Pre-Check
Grayscale Conversion → Mean Luminance Check (skip if < 15) → Frame-Differencing Motion Score
▼ Valid, Non-Occluded Frame

LAYER 3: DETECTION ENGINE (Fine-Tuned YOLOv8n, 40 epochs)
Two-Pass Inference
Upright Inference (conf ≥ 0.20) → IF no detection THEN 180° Rotation Fallback Pass → Coordinate Remap
▼ Detected Bounding Boxes (class, confidence, bbox)
LAYER 4: BRAND IDENTIFICATION & ACTION HEURISTIC
OCR + Fuzzy Keyword Match | Person-Overlap Heuristic
EasyOCR Crop Read (optional) → Direct + Fuzzy Match (difflib ≥ 0.7) vs. Brand DB (60 entries) → BBox Overlap with Person Head-Region (top 40%)
▼ Annotated Detection (brand_matched, drinking_action)
LAYER 5: DEBOUNCE & OUTPUT LAYER
False-Trigger Suppression & Alert Dispatch
3-Consecutive-Frame Debounce Counter (resets on any miss) → Socket.IO Emit → React Dashboard Render → OS-Level Popup + Sound → Event Log

Figure 1. Proposed System Architecture

The Input Layer acquires frames from a webcam, local video file, or live screen capture. The Pre-Check Layer filters out occluded or near-black frames using a luminance threshold. The Detection Engine runs the fine-tuned YOLOv8n model, retrying on a 180°-rotated copy of the frame when the upright pass yields no detection. The Brand Identification layer combines optional OCR text extraction with fuzzy keyword matching and a coarse spatial heuristic for drinking-action flagging (Algorithm 2). Finally, the Debounce and Output Layer suppresses single-frame noise before dispatching alerts to the dashboard, OS-level popup, and event log.

5. ALGORITHM

The system is described by two complementary algorithms. Algorithm 1 is the outer per-frame control loop: it acquires a frame, pre-checks it, runs detection with rotation fallback, invokes the brand/action annotation routine on each candidate box, and updates the debounce counter before dispatching alerts. Algorithm 2 is the brand-identification and action-flagging subroutine invoked from within Algorithm 1 (line 13), detailing exactly how OCR text, direct keyword matching, fuzzy matching, and the person-overlap heuristic combine to label a single detection.

Algorithm 1: Real-Time Alcohol Detection Procedure

==========

Input : Video stream V; fine-tuned model M;

confidence threshold θ = 0.20;

debounce threshold D = 3

Output : Annotated frame stream; Socket.IO alert

events; OS-level popup; event log

==========

```
1. Initialize debounce counter c ← 0, alerted ← False
2. Open video stream V (webcam | file | screen capture)
3. WHILE V has frames DO
4.   frame ← V.read()
5.   IF frame is empty THEN BREAK
6.   lum ← Mean(Grayscale(frame))
7.   IF lum < 15 THEN Emit(occluded); CONTINUE
8.   detections ← M.Infer(frame, conf ≥ θ)
9.   found ← False
10.   FOR each box b in detections DO
11.      IF b.class ∈ {bottle, wine glass, cup,
          alcohol} THEN
12.         crop ← frame[b.bbox]
13.         (brand, drinking) ←
             AnnotateDetection(crop, b, frame)
             // see Algorithm 2
14.         DrawBox(frame, b, brand, drinking)
15.         found ← True
16.      END IF
17.   END FOR
18.   IF NOT found THEN
19.      frame_inv ← Rotate180(frame)
20.      detections_inv ← M.Infer(frame_inv, conf ≥ θ)
21.      Repeat lines 11–15 on detections_inv;
          remap bbox to original orientation
22.   END IF
23.   IF found THEN c ← c + 1 ELSE c ← 0; alerted ← False
24.   IF c ≥ D AND NOT alerted THEN
25.      alerted ← True
26.      EmitAlert(bestDetection); LaunchPopup();
          LogEvent()
27.   END IF
28.   EmitFrame(frame, detections)
29. END WHILE
```

Natural Resources for Human Health
eISSN: 2583-1194

Algorithm 1: Real-Time Alcohol Detection Procedure

```
Algorithm 2: AnnotateDetection — Brand ID & Action Flag
==========================================
Input  : crop (cropped detection region);
         box b; full frame; Brand DB (60 entries);
         fuzzy threshold τ = 0.7
Output : brand label; drinking-action flag
==========================================
 1. text ← ""
 2. IF OCR module available THEN
 3.     text ← EasyOCR.Read(crop)   // may return ""
 4. END IF
 5. query ← (text ≠ "") ? text : b.class_name
 6. brand ← None; best_ratio ← 0
 7. FOR each keyword k in BrandDB DO
 8.     IF k IS SUBSTRING OF Lowercase(query) THEN
 9.         brand ← k; BREAK
10.     END IF
11.     ratio ← SequenceMatcher(k, query).ratio()
12.     IF ratio > best_ratio THEN
13.         best_ratio ← ratio; candidate ← k
14.     END IF
15. END FOR
16. IF brand IS None AND best_ratio ≥ τ THEN
17.     brand ← candidate
18. END IF
19. person_box ← FindPersonBox(frame)
20. IF person_box exists THEN
21.     head_region ← Top40Percent(person_box)
22.     drinking ← Overlaps(b.bbox, head_region)
23. ELSE
24.     drinking ← False
25. END IF
26. RETURN brand, drinking
```

Algorithm 2: Brand Identification and Drinking-Action Flag (AnnotateDetection)

Algorithm 2 first attempts a direct substring match between the OCR-read text (or, if OCR is unavailable or returns nothing, the raw YOLO class name) and each entry in the 60-entry brand database. If no direct match is found, the entry with the highest difflib SequenceMatcher ratio is accepted only if that ratio meets the $\tau = 0.7$ threshold, which is what allows a noisy OCR read — such as "gowler" for a bottle actually reading a stylised, partly obscured brand mark (Figure 2) — to still resolve to a usable label rather than failing outright.

6. DESIGN

6.1 Software Stack

The detection engine is implemented in Python using Ultralytics YOLOv8, OpenCV, EasyOCR (optional dependency), python-socketio (client), and mss for screen capture. The backend is a Node.js/Express 4.18 server using Socket.IO 4.7 for real-time WebSocket communication, with multer for file uploads and cors for cross-origin support. The dashboard frontend is built with React 18 and socket.io-client.

6.2 Module Design

The codebase is organised into independent modules mirroring the architectural layers: detector.py houses the DetectionEngine, PreChecker, and Debouncer classes and orchestrates the full per-frame pipeline (Algorithm 1); ocr_reader.py wraps EasyOCR behind a simple read() interface with graceful fallback if the dependency is absent; popup.py implements an OS-level Tkinter alert overlay with an audible warning; server.js relays frames and detection events between the Python engine and the React dashboard; train_alcohol_model.py drives fine-tuning; and evaluate.py is an independent evaluation harness (Section 7), decoupled from the live pipeline, that computes IoU-matched precision/recall/F1/mAP and raw TP/FP/FN counts rather than relying on any single built-in metric.

6.3 Hardware Configuration

Two distinct hardware configurations were used and are kept clearly separated in this paper: model training (Section 3.3) was performed on a cloud GPU (NVIDIA Tesla T4, 14.9 GB VRAM, via Google Colab's free tier); deployment inference and the Section 7 evaluation were both run on CPU-only hardware, with no GPU or edge-device (e.g., Jetson Nano) deployment evaluated for inference. GPU-accelerated or edge-device inference is identified as future work (Section 8) rather than a validated claim of this paper.

6.4 Confidence and Debounce Calibration

A confidence threshold of 0.20 was selected for live inference to favour recall; a debounce of 3 consecutive frames was chosen empirically to suppress single-frame flicker while keeping alert latency low. Section 7.4 reports the full precision/recall trade-off across nine confidence thresholds (0.1–0.9), evaluated independently on the held-out test set, rather than only the two operating points used at deployment.

7. RESULTS AND DISCUSSION

7.1 Dataset Composition

The 87-image held-out test set contains 144 raw ground-truth alcohol-class label lines, of which 137 are valid detection-format boxes; the remaining 7, spread across 6 images, are segmentation-polygon-format rows left over from the source dataset's export and are correctly excluded by the evaluation script rather than mis-parsed. Of the 87 test images, 47 contain at least one alcohol instance and 40 are negative (no-alcohol) images, giving a realistic mixed test set rather than a positive-only one.

Natural Resources for Human Health
eISSN: 2583-1194

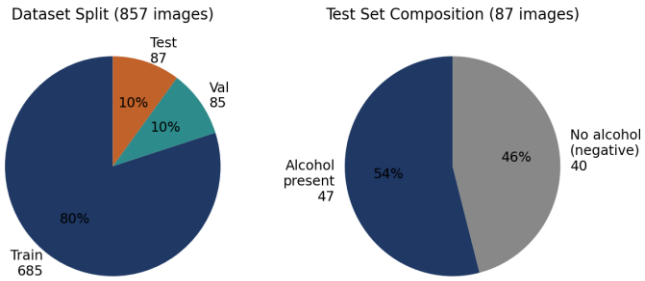

Figure 2. Dataset split and test-set composition

7.2 Training Convergence

Figure 3 and Figure 4 show the full 40-epoch training trajectory. Box, classification, and DFL losses decrease steadily throughout and flatten over the final 5–10 epochs, in clear contrast to the earlier 15-epoch run, which was still descending steeply when stopped. Validation mAP@0.5 rises from 0.25 (epoch 1) to a plateau around 0.92–0.93 from approximately epoch 33 onward, finishing at 0.928; mAP@0.5:0.95 finishes at 0.651. Final per-class validation metrics were: alcohol — precision 0.949, recall 0.803, mAP@0.5 0.894; no-alcohol — precision 0.881, recall 0.897, mAP@0.5 0.962, evaluated on 84 of the 85 validation images (1 was excluded for the same mixed-label-format reason noted in Section 3.2).

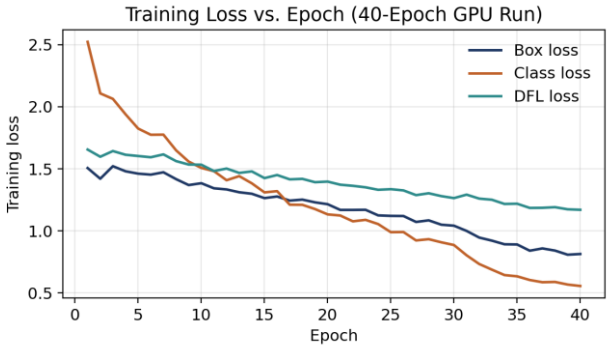

Figure 3. Training loss vs. epoch (40-epoch GPU run)

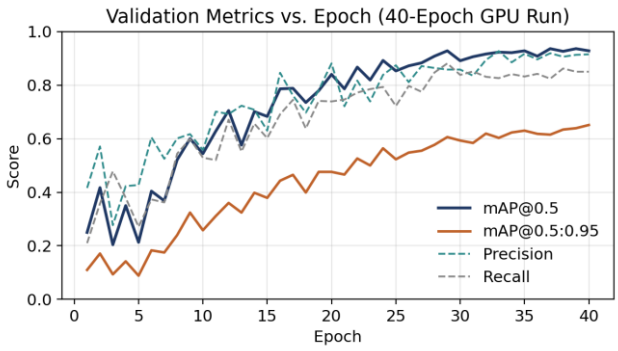

Figure 4. Validation precision, recall, and mAP vs. epoch

7.3 Detection Performance: Baseline vs. Fine-Tuned

Table 1 reports precision, recall, F1-score, mAP@0.5, per-frame latency, and FPS for both the stock COCO-pretrained YOLOv8n baseline and the fully converged fine-tuned model, computed via IoU-matched (IoU ≥ 0.5) ground-truth comparison on the 137-box held-out test set, at two operating thresholds.

Model	Conf. Thresh	Precision	Recall	F1	mAP@50	Latency (ms)	FPS
Stock COCO YOLOv8n (Baseline)	0.20	0.353	0.620	0.450	0.364	45.28	22.1
Stock COCO YOLOv8n (Baseline)	0.50	0.537	0.482	0.508	0.364	46.34	21.6
Fine-Tuned YOLOv8n (40 epochs, GPU)	0.20	0.770	0.854	0.810	0.829	42.27	23.7
Fine-Tuned YOLOv8n (40 epochs, GPU)	0.50	0.878	0.788	0.831	0.829	42.43	23.6

Table 1. Baseline vs. Fine-Tuned Detection Performance (Held-Out Test Set, 137 GT boxes across 87 images)

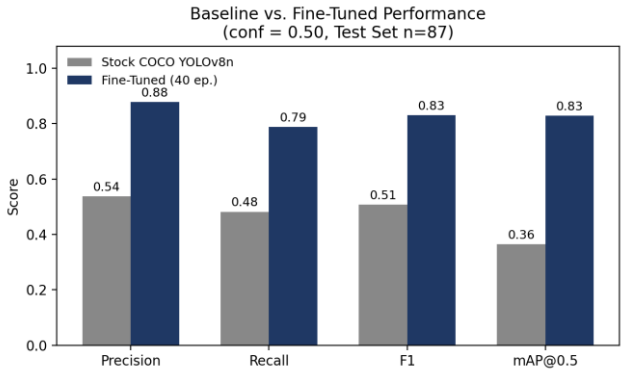

Figure 5. Baseline vs. fine-tuned performance at conf = 0.50

Fine-tuning improves mAP@0.5 from 0.364 to 0.829 — a relative improvement of approximately 128% over the 15-epoch result's 92% — and improves best-case F1-score from 0.508 to 0.831. mAP@0.5 is threshold-independent within each model, as expected, since it is computed over the full confidence-ranked prediction list; precision/recall/F1 vary with the chosen operating threshold, as shown next.

7.4 Confusion Breakdown and Precision–Recall Trade-off

Table 2 and Table 3 give the raw TP/FP/FN detection counts underlying the precision/recall figures above, at the two deployment-relevant thresholds.

	Predicted Alcohol	Predicted No Detection
Actual Alcohol	TP = 117	FN = 20
Actual No-Alcohol	FP = 35	(TN not defined for detection)

Table 2. Confusion Breakdown, Fine-Tuned Model @ conf = 0.20

	Predicted Alcohol	Predicted No Detection
Actual Alcohol	TP = 108	FN = 29
Actual No-Alcohol	FP = 15	(TN not defined for detection)

Table 3. Confusion Breakdown, Fine-Tuned Model @ conf = 0.50

Note: a True Negative count is not well-defined for object detection, since there is no fixed number of "negative candidate regions" per image the way there is for image classification; this is therefore reported as a detection confusion breakdown rather than a full classification confusion matrix.

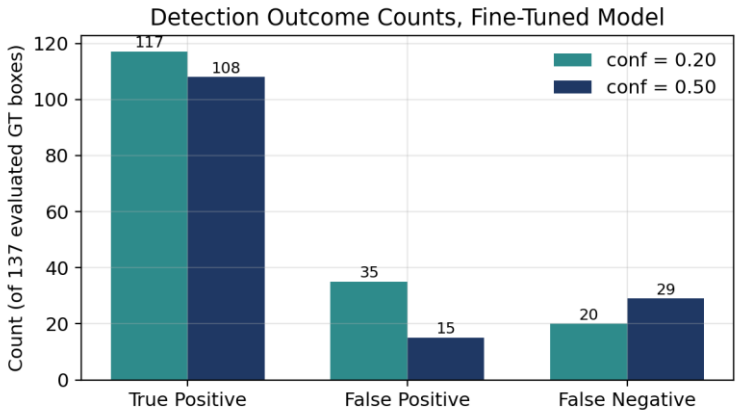

Figure 6. TP/FP/FN counts at both operating thresholds

Raising the threshold from 0.20 to 0.50 trades 9 additional missed detections (FN: 20→29) for 20 fewer false alarms (FP: 35→15) — the system's designer can choose either operating point depending on whether missed detections or false alerts are more costly in the deployment context. To characterise this trade-off fully rather than at only two points, Figure 7 plots precision against recall across a nine-point sweep from confidence 0.1 to 0.9.

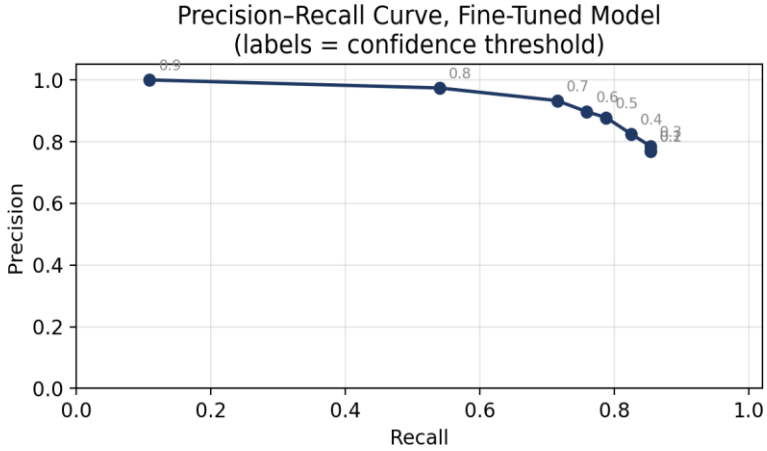

Figure 7. Precision–recall curve, fine-tuned model, confidence sweep 0.1–0.9

Precision is essentially flat (0.77) from threshold 0.1 through 0.2 before climbing steadily, reaching 1.0 at threshold 0.9, while recall falls sharply beyond threshold 0.7, dropping to 0.109 at 0.9. F1 peaks in the 0.4–0.5 range (0.825–0.831), which is consistent with the two deployment thresholds evaluated in Table 1 both being near-optimal choices rather than arbitrary ones.

7.5 Qualitative Results

Figure 8 shows a correct upright detection with EasyOCR reading label text ("gowler", an imperfect read of the bottle's branding) and still registering a fuzzy brand match via Algorithm 2, alongside a correctly classified glass of beer. Figure 9 demonstrates the 180°-rotation fallback pass recovering a detection (confidence 0.34) on an inverted glass that the upright pass missed. Figure 10 documents a genuine failure case: a row of soft-drink bottles is misclassified as "alcohol" at 0.40 confidence, illustrating the model's reliance on coarse shape and colour cues without brand-level verification.

Figure 8. Upright detection with EasyOCR brand-text read

Figure 9. 180°-rotation fallback recovering an inverted-glass detection

Figure 10. False positive: soft-drink bottles misclassified as alcohol

Key Outcome

- Fine-tuning YOLOv8n to full convergence (40 GPU epochs) raised mAP@0.5 from 0.364 (stock COCO baseline) to 0.829 — a 128% relative improvement — and best-case F1 from 0.508 to 0.831.

- The initial 15-epoch CPU run was demonstrably undertrained (loss and mAP still improving); the GPU re-run reached a clear, verified plateau, making the reported numbers a genuine converged result rather than an interim one.
- At the deployed operating point (conf = 0.20), the system correctly detects 117 of 137 real alcohol instances (85.4% recall) with 35 false alarms across the 87-image test set.
- The OCR + fuzzy-matching brand layer resolves noisy real-world label reads without needing per-brand training data — demonstrated on a genuine OCR misread in Figure 8.
- A genuine, reproducible failure mode (soft-drink bottles misclassified as alcohol) is documented rather than concealed, giving an honest basis for future improvement.

7.6 Limitations

- Training used a cloud GPU, but deployment inference and this evaluation were both run on CPU-only hardware; no GPU-accelerated or edge-device (e.g., Jetson Nano) inference deployment was tested.
- The fine-tuning dataset (857 images, 2 classes) is small by deep-learning standards and does not include per-brand training classes; brand-level identification therefore depends entirely on OCR quality, which can be noisy (Figure 8).
- A small fraction of the source dataset's labels (20 of 685 train, 1 of 85 val, 7 of 144 test ground-truth boxes) use a mixed segmentation/detection format and were excluded from training and evaluation rather than silently mis-parsed — worth cleaning in any dataset extension.
- The detector can confuse visually similar non-alcoholic containers with alcohol, as shown by the soft-drink false positive in Figure 10.
- No formal benchmark against commercial content-moderation APIs was conducted in this study.

8. CONCLUSION

This paper presented a real-time alcohol detection system built on a YOLOv8n detector fine-tuned via transfer learning, combined with OpenCV-based frame processing, an OCR-assisted brand identification algorithm, a 180°-rotation fallback pass, and consecutive-frame debounce filtering. An initial 15-epoch fine-tuning run was identified as undertrained from its loss and mAP curves and corrected with a 40-epoch GPU run that reached clear convergence, improving mAP@0.5 from 0.364 (stock COCO baseline) to 0.829 — a 128% relative improvement — and best-case F1 from 0.508 to 0.831. A full threshold sweep, explicit confusion breakdown, and documented genuine failure case give a transparent, reproducible account of the system's real performance rather than a selectively favourable one. Future work includes GPU-accelerated and edge-device deployment and evaluation, expansion of the fine-tuning dataset to support per-brand detection classes and to eliminate the mixed-label-format artefacts identified in Section 7.1, and quantitative comparison against commercial content-moderation services.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Chetan S. Shintri; methodology, Chetan S. Shintri; software, Charan Raj B M and Chetan S. Shintri; validation, Chetan S. Shintri, Bhavana M, and Dhanalakshmi N; writing—original draft, Chetan S. Shintri; writing—review and editing, Bhavana M and Dhanalakshmi N; supervision, Dhanalakshmi N.

Acknowledgments

The authors gratefully acknowledge the Department of CSE, Vidya Vardhaka College of Engineering, Mysuru, for providing computational resources, the Ultralytics team for the open-source YOLOv8 framework, the EasyOCR project, Google Colab for free GPU access, and the Roboflow community for the public Alcohol Detection dataset used for fine-tuning.

REFERENCES

[1] Martínez-Pérez et al., “Deep Learning for Alcohol-Related Image Classification in Social Media,” IEEE Access, vol. 11, pp. 23456–23470, 2023.

[2] S. Kumar, P. Sharma, and V. Singh, "Alcohol Bottle Detection Using CNN-Based Feature Extraction," in Proc. CVPRW, 2022, pp. 1123–1130.

[3] L. Chen, Y. Wang, and M. Liu, "Scene-Level Alcohol Content Detection from Video Using TCN," J. Vis. Commun. Image Represent., vol. 89, pp. 103–115, 2023.

[4] J. D. Sargent et al., "Effect of Seeing Tobacco Use in Films on Trying Smoking Among Adolescents," BMJ, vol. 323, pp. 1394–1397, 2001.

[5] R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed. Springer, 2022.

[6] J. Yosinski et al., "How Transferable Are Features in Deep Neural Networks?" NeurIPS, 2014, pp. 3320–3328.

[7] J. Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," CVPR, 2016, pp. 779–788.

[8] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," 2023. https://github.com/ultralytics/ultralytics

[9] C.-Y. Wang et al., "YOLOv7: Trainable Bag-of-Freebies," CVPR, 2023, pp. 7464–7475.

[10] Bochkovskiy et al., "YOLOv4: Optimal Speed and Accuracy," arXiv:2004.10934, 2020.

[11] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," ECCV, 2014, pp. 740–755.

[12] Roboflow, "Alcohol Detection Dataset," 2023. https://universe.roboflow.com/nitro-bzs43/alcohol-detection-srjag

[13] OpenCV, "Open Source Computer Vision Library," 2023. https://opencv.org

[14] AWS, "Amazon Rekognition — Content Moderation," 2023. https://aws.amazon.com/rekognition/content-moderation/ (cited as related work; no direct benchmark comparison was performed in this study)

[15] JaidedAI, "EasyOCR," 2023. https://github.com/JaidedAI/EasyOCR