FPT UNIVERSITY

SUBJECT:
AI Development with TensorFlow (DAT301m)

PROJECT REPORT

Real-time Spotify Controller via Hand Gesture Recognition using YOLO

LECTURER: Ha Anh Vu

MEMBERS:
Nguyen Van Anh Duy - SE181823
Tran Huynh Bao Minh - SE193028
Tran Hoang Tuan Hung - SE193182
Phan Quoc Anh - SE193726
Tran Quoc Huan - SE193642

CLASS: AI1803

March 22, 2026

Abstract

This paper presents a robust, real-time hand-gesture recognition system designed for media control on macOS environments using the YOLOv8 object detection architecture. The system is trained to detect seven distinct gesture classes—including a critical background-style `no_gesture` class—and maps six active gestures to specific media commands: play, pause, next track, previous track, volume up, and volume down. The training pipeline incorporates a reproducible runtime configuration, strict data split validation, and a 100-epoch fine-tuning regimen leveraging `yolov8n` pretrained weights. Experimental results demonstrate exceptional detection capabilities, achieving a peak mAP@0.5:0.95 of 0.876, with precision and recall scores of 0.991 and 0.990, respectively. To bridge the gap between theoretical accuracy and practical usability, the deployed system integrates a confidence-based top-gesture selection algorithm, temporal stability checks, and action cooldown heuristics. These post-processing strategies significantly mitigate false positive command triggers, resulting in a seamless and reliable touchless human-computer interaction experience in live webcam streams.

Keywords: YOLOv8, Hand Gesture Recognition, Real-Time Computer Vision, Human-Computer Interaction (HCI), macOS Media Control, Object Detection.

Contents

1 Introduction

The evolution of Human-Computer Interaction (HCI) has increasingly leaned towards touchless interfaces, driven by the need for convenience, accessibility, and intuitive user experiences. Vision-based hand gesture recognition serves as a highly practical alternative to traditional keyboard, mouse, or touch inputs, particularly in scenarios where physical contact with the device is undesirable or inconvenient.

Recent advancements in deep learning, specifically in single-stage object detectors like YOLO (You Only Look Once) [3], have enabled real-time, high-accuracy inference on edge devices. In this work, we propose and implement a real-time gesture control framework utilizing a YOLOv8 [2] detector to identify hand gestures from a live webcam stream and execute corresponding media control commands on macOS.

The core contributions of this paper are two fold:

1. **End-to-End Pipeline Optimization:** We detail a complete pipeline from dataset curation and model fine-tuning to deployment, demonstrating quantitative evaluation metrics that validate the efficacy of YOLOv8n for this specific HCI task.

2. **Robust Deployment Heuristics:** We introduce a command-triggering strategy utilizing temporal smoothing and action cooldowns, which effectively resolves the common issue of erratic false-positive actions in real-world, noisy operational environments.

2 Dataset and Problem Definition

2.1 Detection Classes and Mapping

The recognition task is formulated as a multi-class object detection problem. The model is trained to identify seven specific hand configurations: `fist`, `one`, `peace`, `three`, `four`, `palm`, and `no_gesture`.

During deployment, six of these classes are mapped to executable AppleScript [4] media commands. Crucially, the seventh class (`no_gesture`) serves as a negative or background intent class. By explicitly teaching the model to recognize hands that are not forming a control gesture, the system's susceptibility to accidental triggers during normal human movement is drastically reduced.

2.2 Dataset Composition and Splits

The project utilizes a custom-curated dataset inspired by large-scale resources such as the HaGRID (HAnd Gesture Recognition Image Dataset) [1]. Rigorous quality control was applied to ensure diverse lighting conditions, varied backgrounds, and different hand morphologies. The dataset is strictly partitioned to prevent data leakage:

- **Images:** 8,487 for training and 2,121 for validation (Total: 10,608).

- **Bounding Boxes:** 10,467 for training and 2,606 for validation (Total: 13,073).

2.3 Class Distribution

Maintaining class balance is critical for preventing model bias. The six active command classes are relatively balanced in the training set (ranging from 1,372 to 1,444 instances each). The `no_gesture` class is intentionally over-represented (1,981 training instances) to provide stronger regularization and penalization against false positives during live inference. A similar balanced distribution is maintained in the validation subset.

3 Proposed Methodology

3.1 Model Architecture and Environment

To satisfy the strict latency requirements of real-time HCI, the nano variant of YOLOv8 (`yolov8n.pt`) was selected as the backbone. It offers an optimal trade-off between computational efficiency and detection accuracy, making it suitable for standard laptop webcams without requiring dedicated AI accelerators during inference. Model training was conducted on a high-performance Tesla V100 GPU (32GB VRAM).

3.2 Reproducible Data Pipeline

A highly reproducible data pipeline was established prior to training. The system automatically generates a `data_runtime.yaml` file based on the dynamic structural layout of the dataset, performing strict existence checks for training and validation directories. This dynamic path resolution eliminates stale-path errors, ensuring seamless transitions between local development environments and remote GPU servers.

3.3 Training Configuration

The model was fine-tuned for 100 epochs utilizing Stochastic Gradient Descent (SGD). To stabilize the initial learning phase, a 3-epoch warmup period was implemented, transitioning into a Cosine Annealing learning rate schedule. Automatic Mixed Precision (AMP) was enabled to accelerate training and reduce VRAM consumption. The detailed hyperparameters are outlined in Table 1.

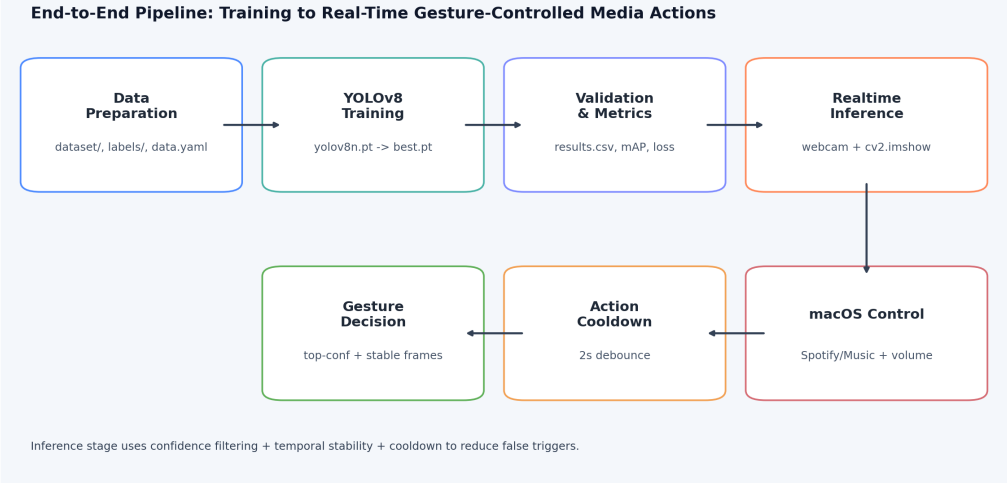

Figure 1: System pipeline overview: Data preparation → YOLOv8 training → Real-time inference → macOS command execution.

4 Experimental Results

4.1 Learning Curve and Convergence

The training process over 100 epochs (logged comprehensively in `results.csv`) demonstrates stable convergence. Early epochs showed expected high loss and moderate precision, but the model rapidly adapted to the specific hand morphology of the dataset.

Table 1: Training Hyperparameters (YOLOv8n Fine-Tuning)

Parameter	**Value**	**Description & Rationale**
Initialization	`yolov8n.pt`	Transfer learning base to accelerate convergence and feature extraction.
Epochs	100	Total dataset passes; optimal for balancing learning without overfitting.
Image Size	640×640	Standard YOLO resolution balancing real-time inference speed and spatial detail.
Batch Size	64	Number of images processed simultaneously, maximizing Tesla V100 GPU utilization.
Workers	8	CPU threads allocated for continuous background data loading and augmentation.
Optimizer	SGD	Stochastic Gradient Descent for stable and reliable weight updates.
Initial LR (lr0)	0.01	Starting learning rate (step size) for early rapid gradient descent.
Final LR Factor	0.01	Reduces final epoch LR to 1% of initial LR for fine-grained convergence.
Momentum	0.937	Accelerates SGD in the optimal direction and dampens oscillations (local minima avoidance).
Weight Decay	5×10^{-4}	L2 regularization penalty applied to weights to prevent model overfitting.
Warmup Epochs	3.0	Gradual LR increases at the start to prevent early training instability (gradient explosion).
Cosine LR	Enabled	Smooths the learning rate decay into a cosine curve for better optimization.

4.2 Metric Analysis at Key Milestones

Table 2 and Table 3 present the progression of critical evaluation metrics. At Epoch 1, the model achieved a baseline mAP@0.5:0.95 of 0.684. The model reached its peak performance at Epoch 85, achieving near-perfect Precision (0.991) and Recall (0.990), alongside a stellar mAP@0.5 of 0.993 and mAP@0.5:0.95 of 0.876.

By the final epoch (Epoch 100), the model maintained these high metrics, with training box loss dropping to 0.518, indicating excellent bounding box localization.

Table 2: Primary Detection Metrics Across Training Milestones

Epoch Milestone	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Epoch 1	0.8032	0.7774	0.8618	0.6841
Best Epoch (85)	0.9909	0.9899	0.9934	0.8762
Final Epoch (100)	0.9949	0.9815	0.9934	0.8734

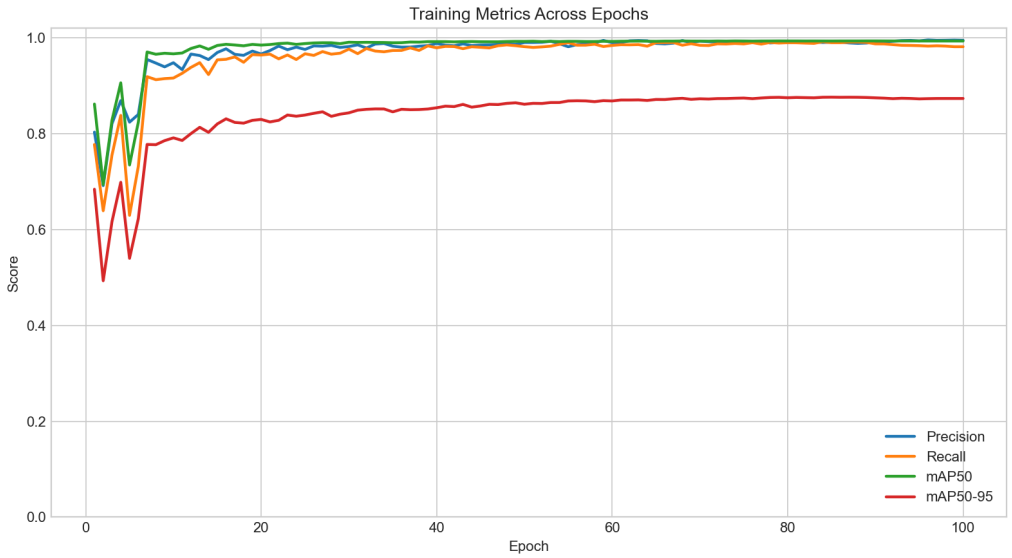

Figure 2: Line graphs illustrating Precision, Recall, mAP@0.5, and mAP@0.5:0.95 trends across 100 epochs.

Table 3: Loss Progression Across Training Milestones

Epoch Milestone	Train Box Loss	Train Cls Loss	Val Box Loss	Val Cls Loss
Epoch 1	1.0079	2.9240	0.7912	1.3077
Best Epoch (85)	0.5848	0.3389	0.5901	0.2474
Final Epoch (100)	0.5187	0.2339	0.5948	0.2492

4.3 Quantitative Improvements

The fine-tuning process yielded an absolute mAP@0.5:0.95 gain of 0.192 from baseline to peak (a relative improvement of approximately 28.1%). The minimal disparity between training and validation loss at the best epoch confirms that the model generalized exceptionally well without overfitting, making it highly reliable for unseen live inference.

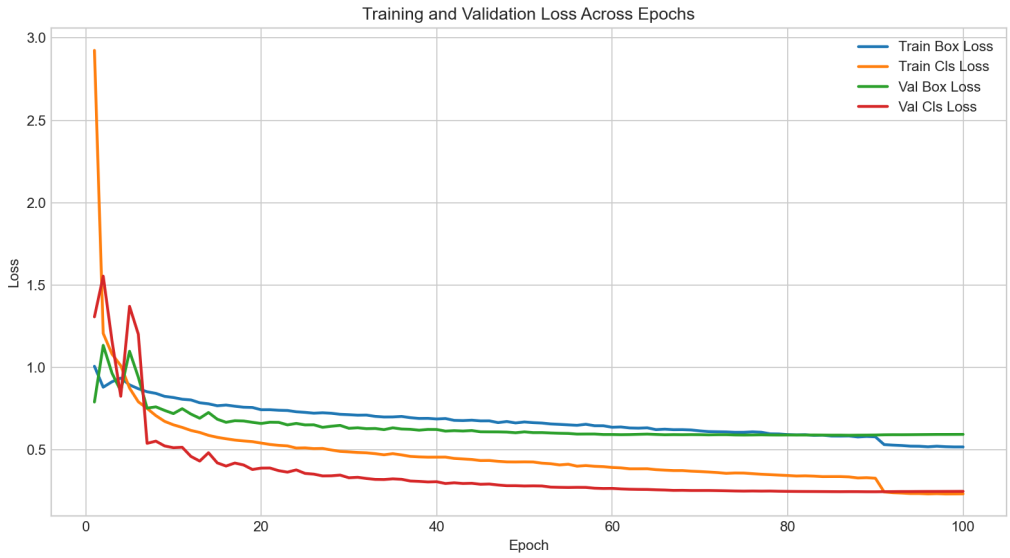

Figure 3: Training vs. Validation Loss curves for bounding box and classification tasks.

5 System Integration and Real-Time Deployment

5.1 Real-Time Inference Pipeline

The model was integrated into a Python-based deployment script using OpenCV. The system captures frames via the user's webcam (utilizing native macOS AVFoundation backends via OpenCV), processes them through the trained YOLOv8 engine, and overlays bounding boxes, confidence scores, and current active states in a real-time GUI window.

5.2 Semantic Command Mapping

The system leverages macOS AppleScript integrations to execute Spotify/Media system controls based on the recognized gestures:

- `fist`: Pause
- `palm`: Play
- `one`: Next Track
- `peace`: Previous Track
- `three`: Volume Up
- `four`: Volume Down

5.3 Temporal Stability and Cooldown Heuristics

A raw object detector operating at 30+ FPS will inherently produce micro-fluctuations in bounding box classifications, potentially causing chaotic media control behavior (e.g., skipping multiple tracks in a single second). To guarantee an enterprise-grade user experience, three deployment heuristics were implemented:

1. **Confidence Gating & Top-1 Selection:** Only the detection with the highest confidence score in a given frame is considered, ignoring background noise.

2. **Temporal Frame Persistence:** A specific gesture must be consistently recognized for a predefined number of consecutive frames (N) before a command is authorized.

3. **Action Cooldown (T_{cool}):** Once a command is fired, a strict temporal cooldown is initiated, during which no further commands are accepted.

This logic successfully transforms raw, frame-by-frame AI detections into deliberate, human-intended actions.

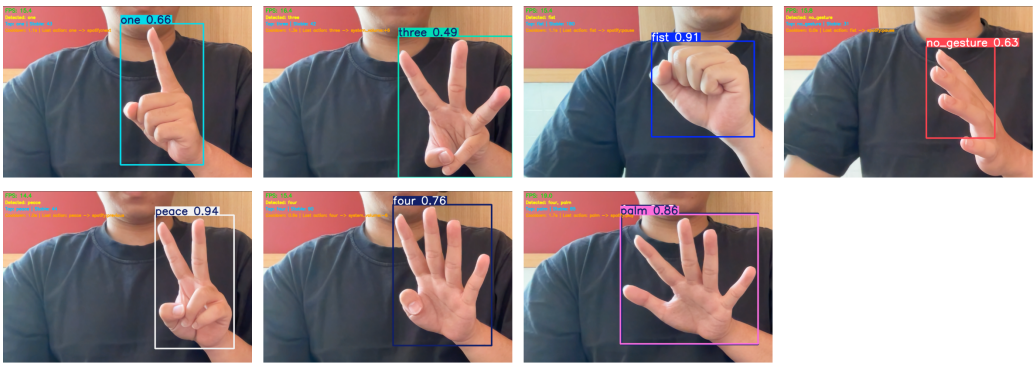

Figure 4: Screenshot of the real-time application GUI showing webcam feed, bounding boxes, FPS counter, and current media action status.

6 Discussion

The empirical results confirm that lightweight architectures like YOLOv8n can achieve near-saturated accuracy metrics (mAP@0.5 > 0.99) on constrained HCI tasks. The deliberate inclusion of the `no_gesture` class proved to be a critical design choice, acting as a buffer that prevents ambiguous hand positions from being falsely classified as actionable commands.

While the system performs robustly under standard conditions, some limitations remain. Performance degrades slightly in extreme low-light environments or under heavy motion blur. Additionally, while overall mAP is high, per-class Average Precision (AP) analysis in future iterations will help identify if specific classes (e.g., `three` vs. `four`) suffer from higher mutual confusion.

7 Conclusion and Future Work

This paper detailed a comprehensive framework for a real-time, gesture-based Spotify controller on macOS. By systematically curating a balanced dataset, fine-tuning a YOLOv8n model, and implementing rigorous post-processing heuristics (temporal persistence and cooldowns), we achieved an optimal balance between high detection accuracy and practical usability. The system effectively translates visual intent into physical OS-level commands without the severe false-positive rates that typically plague raw computer vision pipelines.

Future work will focus on: (1) Generating detailed confusion matrices and per-class AP reports; (2) Exporting the model to ONNX or CoreML to leverage Apple Silicon Neural Engines for lower power consumption; and (3) Expanding the dataset to include complex occlusions and variable lighting to further enhance model robustness.

References

[1] A. Kapitanov, A. Kvanchiani, A. Nagaev, R. Brooks, and E. Antonov, “HaGRID - HAnd Gesture Recognition Image Dataset,” *arXiv preprint arXiv:2206.08219*, 2022.

[2] G. Jocher, A. Chaurasia, and J. Qiu, “YOLO by Ultralytics,” 2023. [Online]. Available: `https://github.com/ultralytics/ultralytics`

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[4] Apple Inc., “AppleScript Language Guide,” *Apple Developer Documentation*. [Online]. Available: `https://developer.apple.com/library/archive/documentation/AppleScript`