Real-Time Conducting Tutor Analysis Using Computer Vision and Machine Learning

By
Jeffrey Ernest
CSC 498: Mentored Research
Date of Submission: 5/8/2026
Mentor: Dr. Salgian

ABSTRACT

This paper presents the Real-Time Conducting Tutor, a computer vision–based system designed to support independent conducting practice through immediate, objective feedback. Using pose estimation from a standard laptop camera, the system analyzes conducting gestures in real time and provides visual feedback on beat accuracy, body movement, and hand usage. This work extends the prior system by introducing a machine learning–based beat detector trained on collected conducting data. An LSTM model processes wrist motion frame by frame to identify beat occurrences, and a rolling tempo estimate is displayed during live sessions. The system is optimized for stable performance on consumer hardware, requiring no specialized sensors. These additions bring the system closer to a complete independent practice tool for student conductors

Keywords

music conducting, computer vision, real-time feedback, gesture analysis, music education, machine learning, LSTM, beat detection.

1. INTRODUCTION

Conducting is a complex skill that requires a conductor to maintain a steady tempo, communicate musical ideas through gesture, and remain clear and readable to the musicians watching them. These skills develop over time through practice and instructor guidance. Despite this complexity, conducting pedagogy has changed very little, relying primarily on direct, one-on-one feedback exchanged between a teacher and student during personal lessons.

A major challenge for student conductors arises between these lessons. When practicing independently, students typically rely on mirrors or self-recorded videos. While these tools allow students to observe themselves, they provide only subjective feedback and lack objective evaluation of timing, posture, and gesture clarity. As a result, students may believe their beat patterns or gestures are correct even when subtle issues are present. Repeated practice without correction can reinforce these habits, making them difficult to change later.

The lack of real-time, objective feedback during independent practice creates a significant barrier to effective learning. Conducting requires attention to multiple elements at once – such as timing, posture, and hand movement – which makes accurate self-assessment difficult without external feedback, even when students sense that something is wrong.

This paper presents the Real-Time Conducting Tutor, a desktop application designed to support independent conducting practice using only a standard laptop camera. The system provides immediate, objective visual feedback on conducting technique without requiring specialized

hardware. By describing the system's design and implementation, this paper demonstrates how computer vision can be used to analyze conducting gestures in real time and support more effective practice outside of traditional instruction.

2. RELATED WORK

Research at the intersection of computer vision and musical gesture analysis has explored a range of approaches for recognizing and evaluating conducting gestures. Early systems focused on accuracy through specialized hardware, while more recent work has shifted toward software-based solutions that rely on markerless tracking. Although these newer approaches improve accessibility, many still struggle to provide low-latency feedback suitable for musical training. This section reviews prior work in conducting analysis and gesture-based learning systems, highlighting the limitations that motivate the approach presented in this paper.

2.1 Evolution from Hardware Sensors to Computer Vision

Early computer-assisted conducting systems relied heavily on specialized hardware due to limited computing power and immature vision algorithms. These systems often used depth sensors, motion controllers, or wearable devices to achieve the precision required for musical applications. While effective in controlled environments, they were expensive and impractical for most students, limiting their use outside of research settings. As computing resources improved, researchers began exploring camera-based alternatives that removed the need for dedicated sensors. Lightweight pose-estimation frameworks such as MediaPipe made it possible to track human motion using standard RGB webcams. For example, Khazaei et al. demonstrated that computer vision can support low-latency gesture analysis without specialized hardware [1]. This shift reflects a broader trend in gesture recognition research, where deep learning–based vision models increasingly replace hardware-heavy solutions [2].

2.2 Feedback Methods in Motor Learning

Effective learning systems depend not only on accurate detection but also on how feedback is presented to the user. Prior research in motor learning emphasizes that feedback must be immediate and easy to interpret in order to support skill development. Fonteles and Rodrigues showed that visual feedback can help users better understand musical structure and motion patterns [3]. Other systems have explored alternative feedback modalities. The Chorify system combines pose detection with vibration-based feedback to help users correct movement in dance training [4]. Similarly, gamified systems such as MUSE visualize hand motion in real time to teach basic conducting patterns [5]. These approaches demonstrate that real-time feedback can

improve awareness and engagement, but many focus on isolated gestures or entertainment rather than sustained conducting practice.

2.3 Challenges in Accessibility and Precision

Despite these advancements, a trade-off remains between accuracy and accessibility. Hardware-based systems such as BeatSaver achieve high precision using inertial measurement units (IMUs), but require users to wear sensors or hold specialized equipment [6]. On the other hand, purely vision-based systems are more accessible but often struggle with latency and reduced accuracy at higher tempos. Studies using Kinect-based classifiers have shown that recognition performance can degrade as tempo increases, limiting their usefulness for real conducting practice [7]. This work addresses these challenges by optimizing a MediaPipe-based vision pipeline for real-time use on consumer hardware. Rather than maximizing spatial precision, the system prioritizes consistent timing and low-latency feedback, which are more critical for conducting pedagogy [8].

2.4 Pedagogical Feedback and Real-Time Learning

Research in motor learning consistently shows that skill development depends on tight feedback loops. Delays between an action and its evaluation weaken the connection needed for effective correction. Huang and Onate applied this principle in the Chorify system, using a “virtual mirror” approach to show that real-time pose feedback can support motor learning without requiring full 3D reconstruction [4]. However, conducting presents additional challenges. Identifying the ictus—the exact moment a beat occurs—requires higher temporal precision than many dance or gesture-recognition systems demand. Post-session video analysis, while useful for review, breaks the immediate feedback loop by delivering information only after gestures have already been reinforced. Real-time feedback allows students to adjust their technique as they practice, making correction more effective and preventing poor habits from forming.

2.5 Gap and Contribution of This Work

The current state of conducting analysis systems reveals a critical gap between high-fidelity hardware and accessible software. Existing tools generally fall into two categories: specialized systems using VR controllers, data gloves, or depth sensors that offer precision but remain inaccessible to the average student [6,7], or markerless vision systems that are hardware-agnostic but have historically suffered from latency issues unsuitable for musical timing [1,2]. There is a distinct lack of tools that are simultaneously accessible (requiring only a standard webcam) and temporally responsive. This work addresses this disparity by proposing a real-time,

webcam-based conducting tutor that provides immediate, corrective feedback without the need for specialized peripherals [9].

3. SYSTEM OBJECTIVES AND DESIGN RATIONALE

The primary goal of the Real-Time Conducting Tutor was to create an accessible training tool that supports independent conducting practice through immediate, objective feedback. The system was designed to address two common limitations in conducting education: the lack of feedback during solo practice and the cost or complexity of hardware-based analysis tools. By using a standard webcam and real-time computer vision, the system aims to help students practice foundational conducting skills outside of lessons in a way that is both practical and informative.

3.1 Core Educational Objectives

The Real-Time Conducting Tutor was designed with the goal of providing clear, objective feedback during independent conducting practice. Rather than replicating full instructional evaluation, the system aims to support the development of foundational conducting habits that are commonly addressed in early training. Specifically, the system aims to provide feedback on tempo consistency and beat placement based on a selected time signature and tempo. In addition, it aims to highlight patterns in overall body movement and hand usage during conducting. These areas were chosen because they are central to clear conducting technique and difficult for students to assess accurately on their own. Together, these objectives guided the design of the system's feedback mechanisms and interaction flow. The system is intended to complement traditional instruction by reinforcing basic concepts during practice sessions where instructor feedback is not available.

3.2 User Interface and Feedback Design

The user interface was designed to support effective independent practice. Because conducting requires continuous visual attention, the system aims to present feedback that can be understood quickly without interrupting the conducting process. The interface design follows Shneiderman's Eight Golden Rules of Interface Design, emphasizing consistency, clear feedback, and predictable interaction. Feedback is delivered through simple visual cues that distinguish correct behavior from areas needing adjustment. Timing-related feedback appears briefly so it does not disrupt the natural flow of conducting, while posture- and gesture-related feedback remains visible as long as the issue persists. This approach reflects the different nature of these corrections and aims to provide useful guidance without overwhelming the user. All session

settings, including tempo and time signature, are configured before practice begins. Separating configuration from active conducting helps maintain a clear workflow.

3.3 Performance and Technical Constraints

Because the system operates in real time, performance constraints played a major role in its design. The system aims to run reliably on standard consumer hardware while maintaining smooth visual feedback and consistent timing, without requiring specialized sensors or graphics acceleration. Pose estimation and gesture analysis are computationally demanding, so the system was designed to balance processing cost with responsiveness. Rather than maximizing precision at the expense of delay, the system prioritizes low-latency feedback so that visual and audio cues remain useful during active conducting. Accurate audio timing was treated as a core requirement. Since conducting depends on a close relationship between motion and sound, the system aims to minimize audio latency and maintain consistent alignment between the metronome and gesture analysis. Finally, the system was designed to remain responsive throughout a practice session. By separating visual processing from interface rendering, the design aims to prevent slowdowns or interruptions that could disrupt practice. These constraints informed the architectural decisions described in the following section.

4. System Architecture

The Real-Time Conducting Tutor builds directly on earlier work that analyzed pre-recorded conducting videos to generate objective feedback, such as timing accuracy and gesture metrics. That earlier system focused on post-session analysis, allowing users to review numerical results and visualizations after a conducting performance had already taken place. While effective for evaluating technique, this approach separated feedback from the act of conducting itself.

The current system extends this foundation by adapting the same core analysis ideas for live use. Rather than focusing solely on post-processing recorded video, the architecture was redesigned to support active training sessions in which feedback is delivered while the user is conducting. This shift introduced new requirements related to timing, responsiveness, and interaction, but the underlying goal remained the same: providing objective insight into conducting technique.

Figure 1 provides an overview of the system's modular architecture, illustrating the separation between video capture and pose processing, gesture analysis, audio timing, session control, and interface rendering. This separation enables the system to maintain real-time responsiveness while coordinating multiple concurrent tasks. By isolating computationally intensive pose analysis from timing-critical audio playback and interface updates, the architecture supports continuous feedback during live conducting without disrupting user interaction.

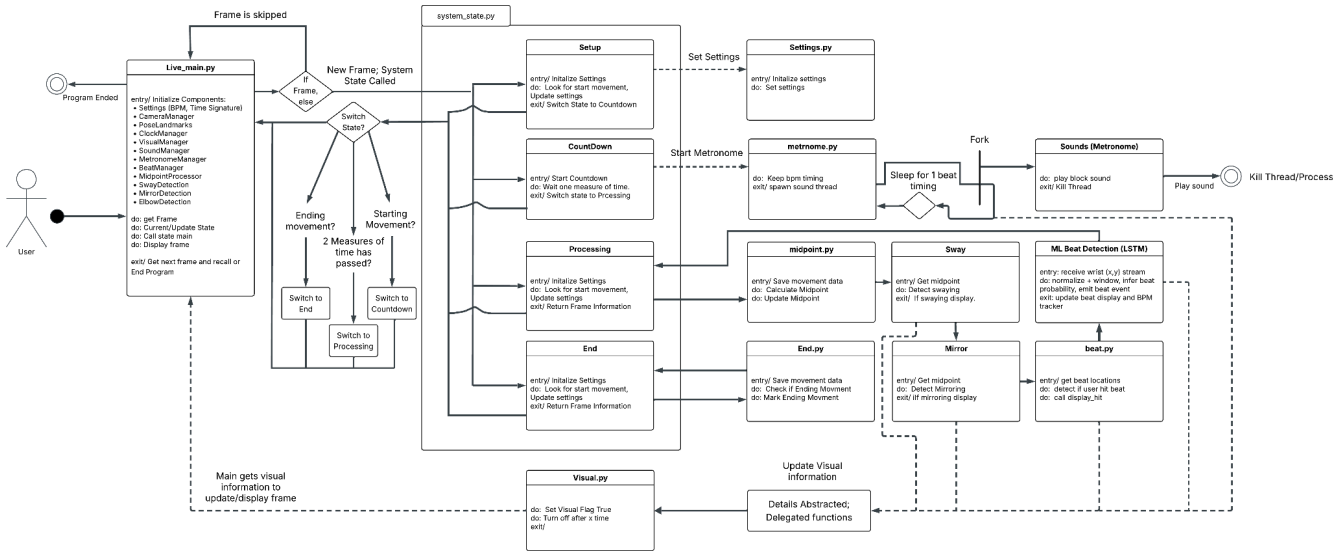

Figure 1: Overview of the live system architecture.

4.1 Legacy Architecture: Video-Based Analysis

The original version of the system analyzed pre-recorded conducting videos and generated objective feedback after a performance had concluded. In this setting, video frames were processed sequentially, allowing the system to extract timing metrics, visual overlays, and gesture summaries without real-time constraints. This approach was effective for post-session evaluation and established the pose-processing and gesture analysis foundation used in this project, as described in prior work [9].

However, because all feedback was delivered after the performance, the system functioned as a diagnostic tool rather than an active training aid. Users could review what their conducting looked like, but could not adjust their technique while conducting. While sufficient for post-processing tasks, this limitation motivated the transition toward a live architecture capable of supporting real-time feedback during practice.

4.2 Transition to Real-Time

Extending the system from video analysis to live conducting introduced new constraints that significantly influenced the system's architecture. While the video-based system could process recordings without regard for time, the live system needed to analyze gestures as they occurred and deliver feedback quickly enough to remain useful during practice. In a real-time setting, delays in video processing, audio playback, or visual feedback directly affect a conductor's sense of timing. Even small amounts of latency can disrupt the relationship between gesture and sound, reducing the effectiveness of feedback. As a result, the system could no longer rely on a sequential processing pipeline and instead required an approach capable of handling multiple tasks at the same time.

This transition also changed the role of feedback. Rather than summarizing a completed performance, the system was redesigned to support active training by responding to gestures while they were being performed. Supporting this shift required coordination between video input, gesture analysis, audio timing, and interface updates, all while remaining stable on standard consumer hardware. These requirements motivated the move to a concurrent architecture that extends the original analysis pipeline to enable continuous, real-time interaction. The following section describes how this architecture supports live conducting practice through parallel processing and separation of system components.

4.3 Live System Architecture

The live version of the Real-Time Conducting Tutor extends the original analysis pipeline to operate continuously during an active conducting session. To support real-time feedback, the system uses a modular, multi-threaded design that separates gesture analysis, audio timing, session control, and interface rendering. Video capture and pose-based gesture analysis run in a dedicated processing thread that continuously evaluates the user's movements. This thread applies the same analytical concepts developed for offline video analysis but adapts them to operate under real-time constraints. Isolating this work ensures that computationally intensive processing does not interfere with timing or feedback delivery.

Rather than attempting to process or retain every frame, the system prioritizes the most recent analysis results, allowing outdated data to be discarded when necessary. This approach favors immediacy over completeness, which is essential for meaningful feedback during live conducting. Session flow is managed through a structured state model that coordinates setup, countdown, active processing, and session completion. This organization ensures that feedback is generated only during intentional conducting and that timing references remain consistent throughout a session. Together, these architectural choices allow the system to deliver continuous, real-time feedback while remaining stable on standard consumer hardware.

4.4 Graphical User Interface

The graphical user interface supports the live architecture by presenting real-time feedback while remaining responsive throughout a conducting session. It functions strictly as a presentation layer and does not perform gesture analysis or timing computation.

The interface integrates both the legacy video-based analysis tools and the live conducting system within a single application, allowing users to select the mode that best fits their practice goals. Figures 2–5 illustrate the full session workflow of the live system, including the home screen (Figure 2), session configuration (Figure 3), real-time feedback during active conducting (Figure 4), and session completion (Figure 5). Together, these views demonstrate how configuration, active feedback, and session termination are clearly separated to support a predictable and focused practice experience.

The interface operates independently from the processing threads and displays the most recent available results. Configuration occurs before practice begins, while feedback is presented only during active conducting, supporting a clear and predictable workflow.

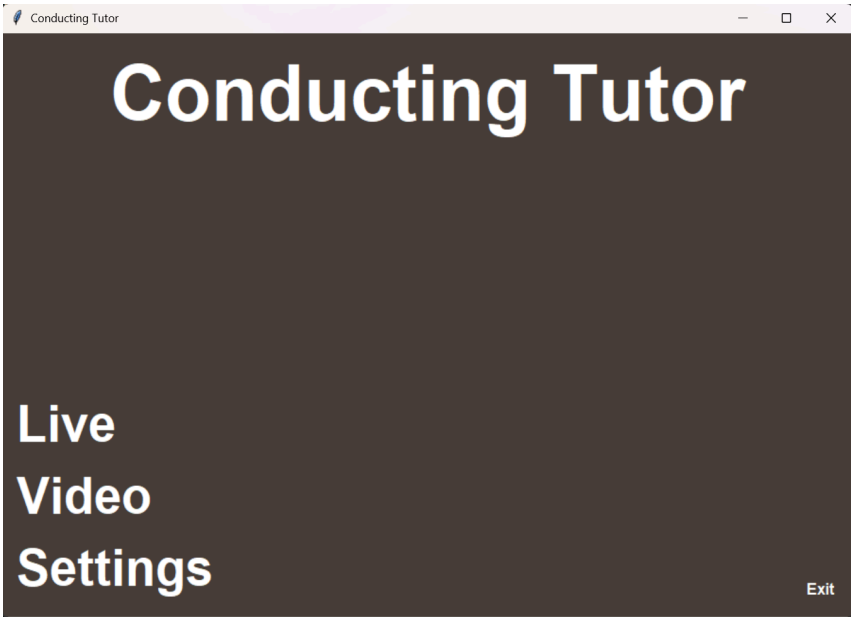

Figure 2: Home screen.

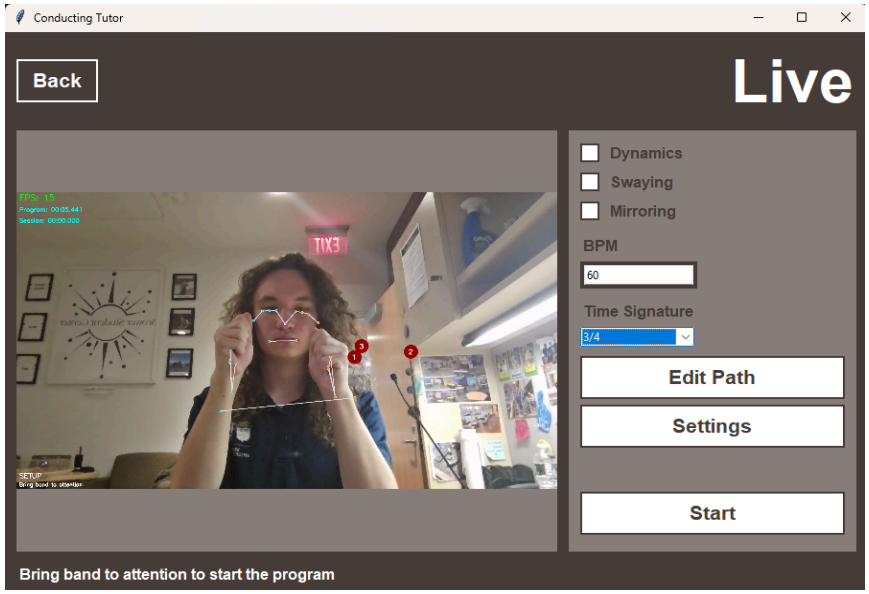

Figure 3: Session configuration screen.

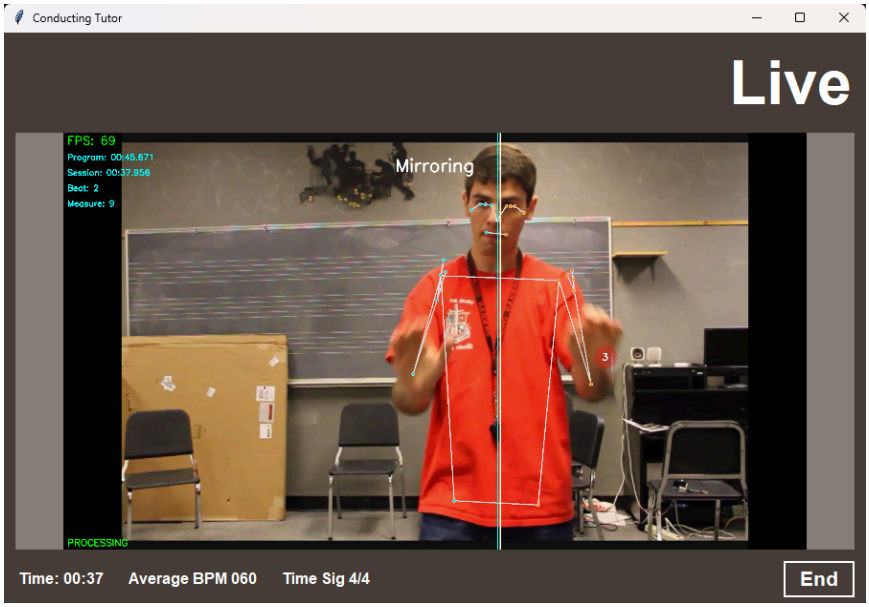

Figure 4: Real-time feedback displayed during an active conducting session.

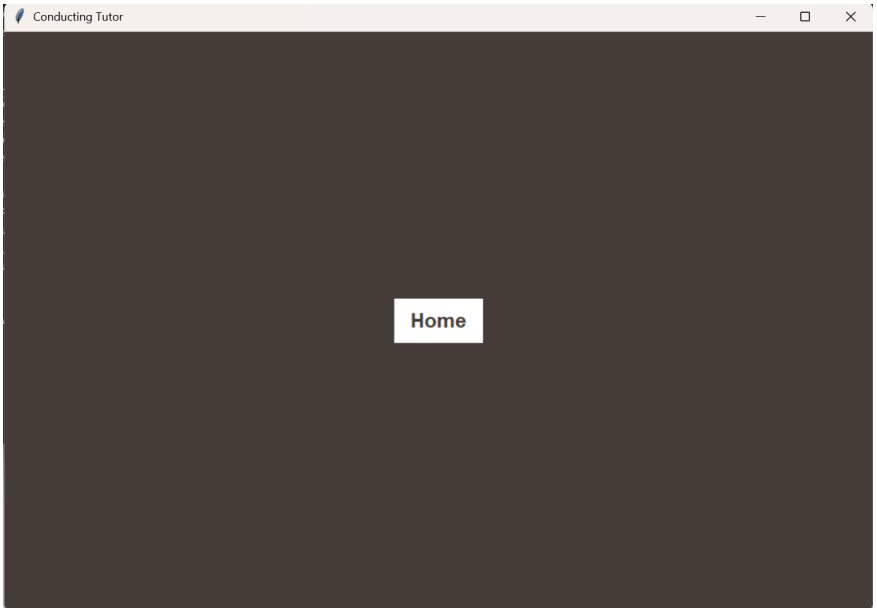

Figure 5: Session completion.

5. METHODOLOGY

The Real-Time Conducting Tutor evaluates conducting technique by continuously analyzing pose data during an active session. All feedback is derived from a shared stream of pose landmarks that are processed once per frame and reused across multiple analysis components. This pipeline-based approach allows posture, gesture, and timing analysis to operate efficiently and consistently in real time.

5.1 Pose Estimation

Pose data is extracted using Google MediaPipe Pose, which provides real-time skeletal landmarks without requiring markers or calibration. MediaPipe returns three-dimensional landmark positions for major body joints, allowing the system to track conducting motion using only video input.

Although full-body landmarks are available, the system focuses on upper-body joints most relevant to conducting, including the shoulders, elbows, wrists, and hips. These landmarks form the foundation for all subsequent analysis. To ensure consistency across users, landmark positions are provided in a normalized coordinate space relative to the camera frame. This allows gesture analysis to remain stable despite differences in body size, distance from the camera, or camera placement, and removes the need for per-user setup. The full set of skeletal landmarks

provided by MediaPipe Pose, including upper-body joints used in this system, is illustrated in **Figure 6.**

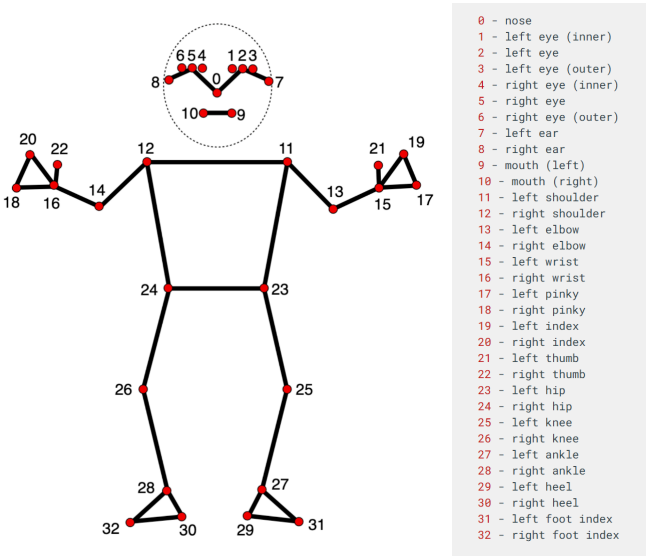

Figure 6: MediaPipe Pose landmark model.

5.2 Posture and Gesture Analysis

Gesture and posture analysis operates directly on the pose landmarks extracted each frame. Several shared reference values are computed early in the processing pipeline and reused across multiple detections to maintain consistency and reduce redundant computation.

One such reference is the midpoint between the left and right shoulders, which serves as a stable indicator of upper-body position. This midpoint is used as a baseline for detecting lateral body movement and provides a shared frame of reference for posture-related analysis. All gesture detections apply temporal filtering, meaning that feedback is only triggered when a behavior persists across multiple frames. This prevents momentary movements, expressive gestures, or pose estimation noise from producing misleading feedback.

5.2.1 Sway Detection

Sway detection evaluates lateral movement of the conductor's upper body. At the start of a session, the shoulder midpoint establishes a neutral reference position. As the session continues, horizontal deviations from this reference are tracked over time. When sustained lateral movement exceeds a predefined threshold, the system interprets this as excessive swaying and

provides corrective feedback. Requiring persistence over multiple frames ensures that natural motion and expressive gestures are not incorrectly flagged.

5.2.2 Hand Mirroring

Hand mirroring is analyzed by comparing the motion patterns of the left and right wrists. Rather than evaluating individual frames, the system examines wrist trajectories over time to determine whether both hands are moving in a highly similar manner. Mirroring feedback is generated only when symmetrical motion persists without an apparent change in intent. This approach allows intentional mirrored gestures while still making users aware of unintentional duplication that may reduce clarity.

5.2.3 Elbow Tracking

Elbow tracking is used to identify inefficient arm motion, particularly gestures where the elbows are raised excessively, resulting in large or uncontrolled movements that resemble arm flapping rather than deliberate conducting. The system evaluates elbow position by examining the geometric relationship between the shoulder, elbow, and wrist landmarks. For each arm, vectors are formed between these joints, and the angle between them is computed using a dot product, providing a stable measure of elbow elevation. When this angle exceeds a predefined threshold, the system flags the gesture and provides visual feedback to the user.

5.3 Beat Configuration and Session Parameters

Before each practice session, the user selects a time signature and tempo. These parameters define the musical context for the session and establish the constraints under which conducting gestures are evaluated. Tempo, specified in beats per minute (BPM), determines the timing reference used for beat evaluation. The selected time signature defines both the number of beats per measure and the expected conducting pattern. Based on these settings, the system generates a corresponding set of virtual beat targets that represent expected ictus locations within the conducting space. By treating time signature and tempo as core inputs rather than interface-level options, the system ensures that gesture analysis and beat detection operate consistently across different musical contexts. This design allows the same analysis logic to be reused for multiple meters and tempi while supporting realistic conducting practice. The resulting virtual beat targets generated from the selected time signature are visualized spatially within the conducting plane, as shown in **Figure 7**.

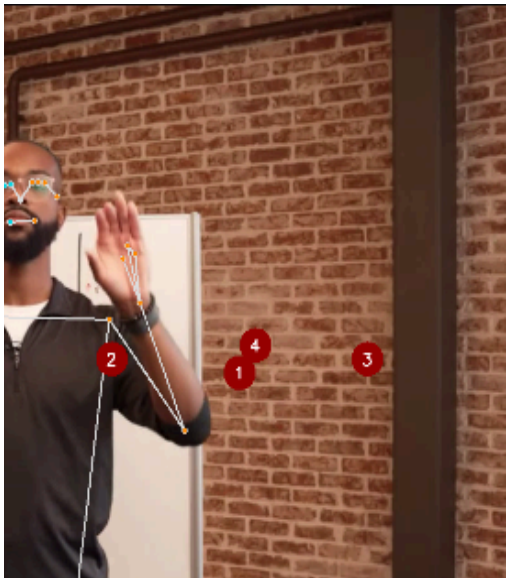

Figure 7: Virtual beat targets generated from the selected time signature.

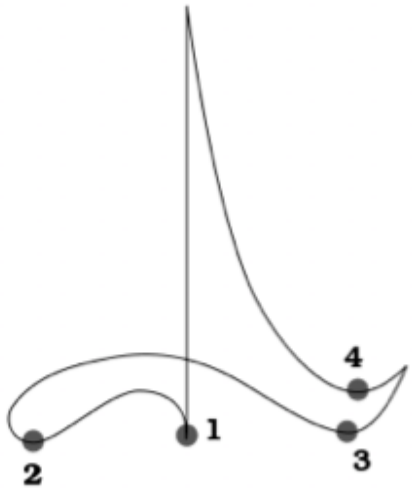

Figure 8: Expected ictus points and conducting path for a standard 4/4 time signature, showing the spatial targets used for beat detection.

5.4 Metronome and Beat Detection

Beat accuracy evaluation requires a stable timing reference that remains consistent throughout a conducting session. The system separates timing control from gesture analysis so that audio playback and beat evaluation remain reliable even when visual processing load varies.

5.4.1 Metronome Timing

The metronome provides a continuous temporal reference for evaluating beat accuracy. Audio playback is handled using the PyDub library, which allows precise control over timing while remaining lightweight enough for real-time use [12]. Rather than starting and stopping audio playback dynamically, the metronome operates continuously and is gated by session state. This design ensures stable timing throughout a conducting session and avoids inconsistencies that can occur when audio playback is repeatedly initialized.

Metronome timing operates independently from gesture analysis and visual processing. By decoupling audio timing from frame-based computation, the system maintains consistent beat references even when processing load varies.

5.4.2 Beat Detection

Beat detection focuses on identifying the ictus, the most decisive point of a conducting gesture. The system supports two complementary approaches that operate in parallel during a live session.

The first approach uses the virtual spatial targets described in Section 5.3. During each frame, the distance between the active wrist and the current beat target is evaluated. When the wrist enters the target region within an acceptable time window around the metronome reference, the position is marked. This method provides a visible spatial reference for where each beat should occur and gives the conductor a clear indication of whether their gesture is landing in the right location at the right time. This spatial marker overlay can be enabled or disabled through the session settings.

The second approach uses a machine learning model that detects beats based on the motion of the conducting hand over time. Rather than checking proximity to a fixed target, the model evaluates a short sequence of wrist positions and determines whether the hand is at a beat point based on the shape of its recent trajectory. This method operates continuously during the Processing phase and produces the primary beat signal used for tempo tracking. The design and training of this model are described in Section 5.6.

5.5 Gesture-Based Session Control

When the user selects Live from the home screen, the system enters the Setup phase immediately and begins watching for a gesture before any analysis starts. This design allows the full conducting workflow to be gesture-driven from that point forward: the session starts, runs, and ends without requiring the user to interact with the keyboard or mouse again. For users who

prefer direct interaction, on-screen buttons are available at each stage as an alternative to the gestures, so the gesture-based flow is optional rather than required.

To begin, the conductor raises both hands upward simultaneously, mirroring the bringing the band to attention gesture a conductor uses to prepare an ensemble before a piece starts. The system detects this when both wrists move upward past a threshold and hold that raised position for two seconds. Once the hold duration is met, the system transitions to the countdown phase automatically. If either hand drops before the two seconds elapse, the timer resets and the system waits for the gesture again. This prevents an accidental or passing arm movement from starting the session unintentionally.

To end a session, the conductor performs a cutoff gesture that mirrors how a conductor signals the end of a piece to an ensemble. The gesture is detected as a four-stage sequence: the wrists cross, then uncross, remain still for one second, and then hold motionless for two additional seconds. Each stage must be completed before the next begins. If the wrists re-cross at any point during the sequence, the system resets and waits for the gesture to restart from the beginning. Jitter-tolerant thresholds are applied to both the crossing check and the stillness requirement so that normal conducting motion that precedes the cutoff does not accidentally trigger it. A user who prefers not to use the cutoff gesture can press the End button at any point to stop the session directly.

Together, these two gestures allow a complete practice session to proceed without interrupting the physical act of conducting. The user can step up to the camera, raise their hands to begin, conduct through the session, and cross their hands to end – all without returning to the keyboard or mouse.

5.6 Machine Learning Beat Detection

The beat detection system was extended to detect when a beat occurs based on the actual motion of the conducting hand rather than its position relative to a fixed spatial target. This required building a training dataset from scratch, selecting a model architecture suited to the problem, and integrating the trained model into the existing live pipeline.

5.6.1 Data Collection

No public dataset of labeled conducting gestures was available, so training data was collected manually using a recording tool built for this purpose. The tool uses the same MediaPipe pose pipeline as the live system to track the right wrist position frame by frame during a recording session. During each recording, the user presses a key to mark each beat as it occurs, generating a binary label per frame – 0 for no beat, 1 for a beat.

A playback and review step was included before saving each session. After recording, the session could be replayed with the labeled beat frames highlighted, allowing the labels to be verified against the actual motion before the data was saved. This step was included because the quality of the training data directly affects what the model learns, and incorrectly labeled frames introduce patterns that are difficult to detect or correct later.

In total, 24 sessions were recorded across the 2/4, 3/4, and 4/4 time signatures at varying tempos. Each session was saved as a CSV file with one row per frame containing the position, velocity, acceleration, and beat label for that frame.

5.6.2 Feature Extraction and Normalization

Before being used for training or inference, raw wrist coordinates are passed through a bounding box normalizer. The normalizer tracks the minimum and maximum x and y values seen across all frames so far and maps each new coordinate into a [0, 1] range within that box. This removes differences caused by conductor position or distance from the camera, so the model receives consistently scaled input regardless of setup.

Two model variants were trained and compared during development: one using all six derived features: x, y, vx, vy, ax, ay and one using only the normalized x and y coordinates. The six-feature model included velocity, computed as the frame-to-frame change in normalized position, and acceleration, computed as the frame-to-frame change in velocity. While the six-feature model achieved higher accuracy on paper, the two-feature model performed well enough in practice and significantly reduced the computation required per frame, making it a better fit for real-time use on consumer hardware. The two-feature model was selected for this reason, and the rest of the pipeline is built around it. The same normalizer logic is used in both the training pipeline and the live inference wrapper, ensuring that the model receives input during live sessions formatted identically to the data it was trained on.

5.6.3 Training Data Preparation

Sequences were extracted from the collected CSV files using an 11-frame sliding window, as illustrated in Figure 9. Each window is labeled as a positive example if it contains a beat frame at positions 1 through 9 within the window, and as a negative example if it contains no beat frames. Placing the beat at multiple positions within positive windows reduces the model's sensitivity to exactly where in the sequence the beat falls, allowing it to generalize across slightly different timing.

After processing all sessions, the dataset contained 2,910 positive and 2,782 negative windows. Approximately 20% of the data was held out for validation, leaving 4,452 samples for training and 1,112 for validation.

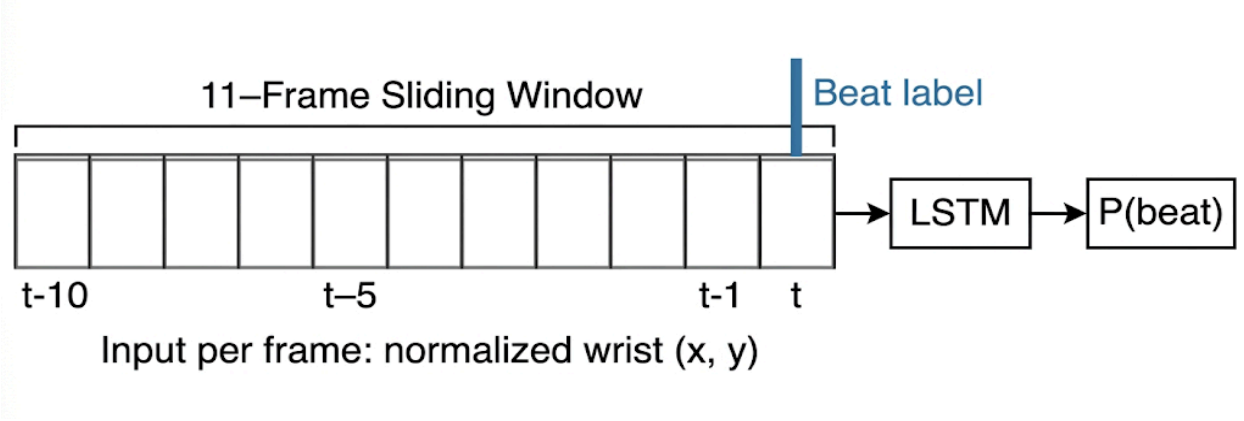

Figure 9: 11-frame sliding window used for sequence construction. Each window passes normalized wrist (x, y) coordinates per frame into the LSTM, which outputs a beat probability.

5.6.4 Model Architecture

The model is an LSTM-based binary classifier built using Keras. LSTM was chosen over a 1D CNN because detecting a conducting beat requires understanding how the hand has moved over a short window of time, not just recognizing a fixed spatial shape. A beat occurs at the reversal point of a downward gesture where the hand stops and changes direction. This only makes sense in the context of the frames leading into it. LSTM processes the sequence one frame at a time and maintains an internal state between steps, allowing it to carry information about prior motion forward as it evaluates each new frame.

The architecture takes an input of shape (11, 2) – 11 frames of normalized x and y coordinates. An LSTM layer with 64 units processes the sequence and produces a summary vector. A dropout rate of 0.3 is applied after the LSTM layer during training to prevent the model from depending too heavily on any individual unit. A dense layer with 32 units and ReLU activation reduces the output further, followed by a second dropout of 0.2. A final dense layer with a single unit and sigmoid activation produces a probability between 0 and 1, where values at or above the detection threshold are treated as a beat.

5.6.5 Training

The model was trained using the Adam optimizer with binary cross-entropy loss. Training ran for 57 epochs before converging, reaching 85.1% accuracy on the training set and 85.0% on the validation set, with training loss of 0.321 and validation loss of 0.342.

During development, two model variants were compared: a six-feature model using position, velocity, and acceleration, and a two-feature model using position only. The six-feature model

achieved higher validation accuracy (85.0%) compared to the two-feature model (77.6%). However, when tested against live video, the two-feature model performed comparably in practice while running significantly faster. The six-feature model introduced enough overhead that it could not maintain real-time frame rates on standard laptop hardware, while the two-feature model ran at near-real-time speeds. The two-feature model was selected for integration based on this tradeoff, consistent with the system's broader priority of stable performance on accessible hardware.

The raw validation accuracy does not fully reflect how the model performs in practice. Two adjustments improved real-world detection significantly. First, rather than requiring an exact frame match, a beat is considered detected if the model fires within a two-frame window around the actual beat. Under this looser timing criteria, accuracy increased to approximately 94%. Second, the detection threshold was raised above the default 0.5, requiring the model to be more confident before registering a beat. This reduced false positives during normal conducting motion without meaningfully increasing missed beats. Together these two tuning choices brought the model's live performance well above what the raw validation numbers suggest.

5.7 BPM Tracking

Once beat detection is active, the timestamps of detected beats are used to estimate the conductor's current tempo. Rather than averaging over the full session, BPM is calculated using a rolling five-second window. This means the displayed value reflects recent conducting speed rather than a session-wide average, allowing the system to respond when a conductor intentionally speeds up or slows down over the course of a piece.

The BPM value is shown as a visual overlay during active conducting and updates approximately every five seconds. If fewer than two beats have been detected in the current window, no value is displayed. This prevents the overlay from showing a misleading estimate at the start of a session before enough data has been collected.

5.8 State Management

System behavior is coordinated using a simple four-phase state model consisting of Setup, Countdown, Processing, and Ending. This structure ensures that configuration, analysis, and feedback occur only when appropriate and in a predictable order. During the Setup phase, session parameters such as tempo and time signature are configured. The Countdown phase provides a brief preparation period before analysis begins. Active gesture analysis and feedback occur only during the Processing phase, ensuring that feedback is tied to intentional conducting behavior. The Ending phase cleanly terminates the session and resets system state. This state-based

structure maintains consistent timing references, prevents premature feedback, and supports a focused practice workflow suitable for independent use.

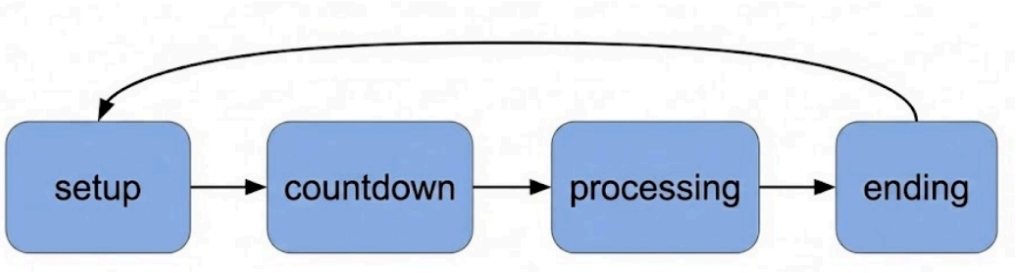

Figure 10: State Management Pipeline.

6. Results & Discussion

The Real-Time Conducting Tutor was evaluated against its primary goals: providing immediate, objective feedback during conducting practice while maintaining stable performance on standard consumer hardware. Overall, the system met these goals and successfully extended prior video-based analysis into a responsive, real-time training tool.

6.1 Performance and Stability

A central requirement of the system was maintaining real-time responsiveness despite the high computational cost of pose estimation. MediaPipe pose extraction is the most expensive operation in the pipeline, and running it on every frame can quickly overwhelm consumer hardware. To address this, the system implements a pose-skipping strategy in which full pose estimation is performed every other frame. For intermediate frames, the system reuses the most recently computed pose data rather than re-running the pose model. This approach significantly reduces processing load while preserving sufficient temporal accuracy for conducting analysis.

In practice, this optimization allowed the system to maintain a stable frame rate of approximately 30 frames per second on standard laptop hardware. Because conducting gestures change gradually from frame to frame, reusing pose data did not produce noticeable degradation in visual feedback or timing accuracy. Separating pose processing, audio timing, and interface rendering further contributed to system stability. During extended practice sessions, the interface remained responsive and timing alignment between gesture analysis and metronome output was preserved. These results demonstrate that real-time conducting analysis using computer vision is feasible on consumer hardware when performance-aware design choices are applied.

Integrating the machine learning model into the live pipeline introduced additional processing overhead. By default, TensorFlow evaluates operations line by line each time the model runs. This caused a noticeable slowdown in frame rate when running inference on every frame. To address this, the prediction function was wrapped with the @tf.function decorator with reduce_retracing enabled. This causes TensorFlow to compile the function into an optimized execution graph on the first call and reuse it for subsequent calls, removing the repeated overhead of converting between Python and TensorFlow data formats. This change significantly reduced per-frame inference time and allowed the two-feature model to maintain stable real-time performance without requiring a GPU.

6.2 Visual Feedback Effectiveness

The system's visual feedback proved effective at communicating conducting accuracy and highlighting areas needing adjustment during live practice. Beat accuracy feedback provided clear confirmation of correct ictus placement while making timing errors immediately visible. The use of spatial tolerance in beat targets allowed for natural variation in gesture shape without penalizing stylistic differences, while still enforcing rhythmic precision. Posture-related feedback, including sway, hand mirroring, and elbow tracking, helped surface issues that are difficult to identify using mirrors or self-recorded video alone. These visual indicators drew attention to sustained patterns rather than isolated movements, allowing users to recognize and adjust problematic habits as they occurred. An example of real-time visual feedback during an active conducting session, including beat confirmation and posture-related cues, is shown in **Figure 11**.

Overall, the visual feedback was easy to interpret at a glance and did not interfere with the act of conducting. Users could remain visually engaged with the screen while making real-time corrections, supporting the system's goal of active, independent practice.

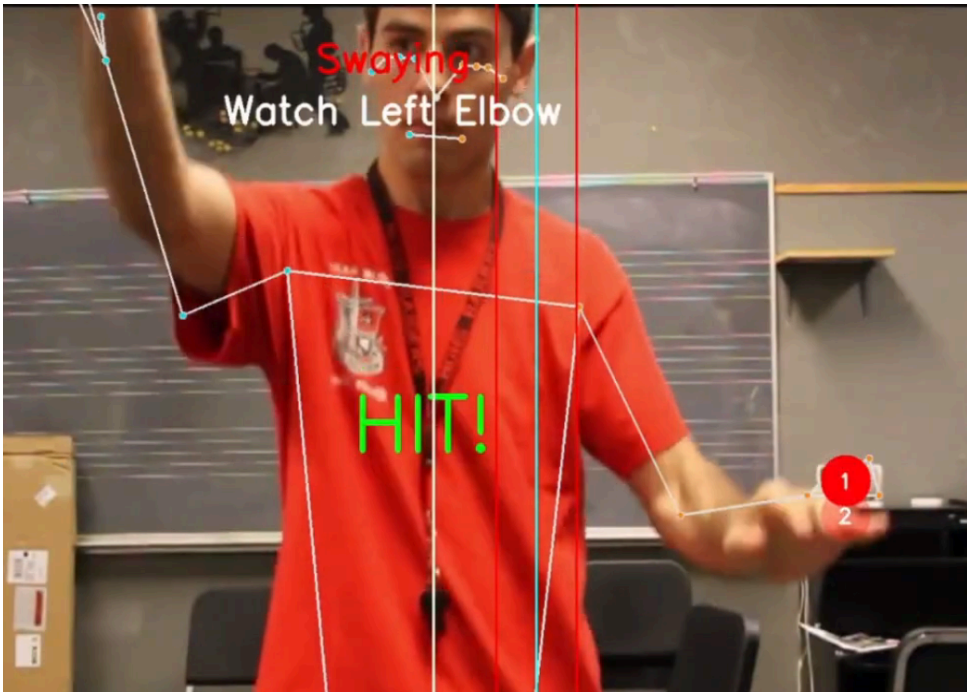

Figure 11: Real-time visual feedback during a live conducting session.

6.3 Validation and Tracking Capabilities

The system was tested across a range of tempi and time signatures to evaluate whether gesture analysis and beat detection remained consistent under different musical conditions. Because tempo and meter are treated as configurable inputs rather than fixed assumptions, the same analysis logic was applied without modification across sessions.

In practice, beat detection and visual feedback remained stable as tempo increased and as time signatures changed. Expected ictus locations updated correctly based on the selected meter, and timing feedback remained aligned with the metronome reference. This confirmed that the system's gesture evaluation is not tied to a specific musical context. Although the system does not yet generate long-term performance summaries, real-time tracking remained reliable throughout individual sessions regardless of configuration. These results suggest that the system can support a variety of conducting practice scenarios while maintaining consistent behavior and feedback.

6.4 Interface Design Evaluation

The interface design was evaluated using Shneiderman's Eight Golden Rules of Interface Design as a guiding framework [10]. Several aspects of the system align well with these principles.

Consistency is maintained through uniform visual feedback, color usage, and interaction patterns across sessions. Informative feedback is provided continuously during active conducting, and clear session boundaries—setup, countdown, processing, and ending—help users understand system state and maintain a predictable workflow.

However, the interface does not fully satisfy all eight rules. In particular, the system provides limited support for error prevention and user confirmation. Actions such as exiting a session or changing configuration settings occur immediately, without confirmation prompts or warnings. While this behavior keeps interaction lightweight, it reduces protection against accidental actions and does not fully support the principle of preventing errors or offering clear closure for user actions.

These tradeoffs reflect a design choice to prioritize uninterrupted practice over additional interaction steps. While the interface effectively supports focused, real-time conducting, future revisions could better align with Shneiderman's framework by introducing confirmation and feedback mechanisms that improve user control without significantly increasing cognitive load.

6.5 ML Beat Detection Performance

The beat detection model was evaluated on the held-out validation set of 1,112 samples. The model achieved 85.0% overall accuracy, with precision and recall balanced closely across both classes: beat precision was 85.6% with a recall of 84.4%, and no-beat precision was 84.6% with a recall of 85.8%. The confusion matrix showed 469 true positives, 477 true negatives, 87 false negatives, and 79 false positives.

These numbers reflect the model's raw performance against exact frame labels. In practice, detection accuracy improved to approximately 94% when evaluated with a two-frame timing tolerance, which accounts for the small natural variation between when a beat is labeled and when the model fires. Raising the detection threshold above the default further reduced false positives during normal conducting motion without meaningfully increasing missed beats. The combination of these two adjustments brought live performance well above what the validation numbers alone suggest.

The close balance between precision and recall indicates the model does not strongly favor one type of error over the other, which is what you want from a tool meant to give unbiased feedback. The remaining error is likely influenced by the fact that all training data was recorded by a single person, limiting the model's exposure to variation in conducting style, body proportion, and camera placement. Despite this, the model performed consistently during live testing without a noticeable rate of false triggers during normal conducting.

6.6 Packaging and Distribution

To move the system from a development environment to something a student could install and run independently, the application was packaged as a standalone Windows executable using PyInstaller [22]. This removed the requirement for users to install Python, manage dependencies, or run the program from a terminal, which is consistent with the system's goal of being accessible to students without a technical background.

The packaging process required resolving several runtime issues that did not appear during normal development. MediaPipe relies on external model files and runtime content that PyInstaller does not include automatically. These missing assets were identified through error output from the packaged build and added to the bundle explicitly. The build environment was also simplified by creating a minimal Conda environment containing only the packages the application required. This reduced bundle size and build time compared to packaging from a full development environment.

The result is a self-contained executable that launches directly into the conducting tutor interface. This moves the project closer to being something a student can simply download and use.

7. Future Work

There are a few directions worth exploring further. One area is continuing to grow the training dataset. Data was collected from YouTube recordings and classroom footage, providing a reasonable range of conducting styles and environments. Additional data would only improve detection accuracy and consistency.

Expanding gesture recognition to include conducting dynamics is another direction. Features such as gesture size and hand velocity could be used to detect when a conductor is signaling a change in volume or intensity, moving feedback beyond timing and structure into expressive technique.

Future versions could also allow users to define custom conducting paths and beat targets tailored to specific pieces or styles. This would support more advanced practice scenarios beyond the standard patterns the system currently recognizes.

Finally, generating a post-session report would add meaningful value to the tool. A summary of metrics such as beat detection results, body sway frequency, and hand mirroring duration would give students a clearer picture of their practice and provide instructors with objective data to supplement traditional lessons.

8. Conclusion

This work presented the Real-Time Conducting Tutor, a computer vision–based system designed to support independent conducting practice through immediate, objective feedback. The system was built to bridge the gap between instructor-led lessons and solo practice by providing corrective feedback on timing, posture, and gesture in real time.

Using pose estimation from a standard laptop camera, the system evaluates beat accuracy, body movement, and gesture behavior while maintaining stable performance on consumer hardware. The gesture-based session control allows a complete practice session to be conducted without interrupting the physical act of conducting – the user raises their hands to begin and performs a cutoff to end, with no keyboard or mouse required. Beat detection was extended beyond fixed spatial targets by training an LSTM model on collected conducting data, enabling the system to recognize beats based on the actual shape of the conducting gesture over time. A rolling tempo estimate provides additional feedback during active sessions, allowing the conductor to monitor their consistency without breaking focus.

Careful architectural design, including concurrent processing, frame-skipping optimization, and compiled inference, kept the system running at stable real-time frame rates without requiring specialized hardware. The results demonstrate that computer vision and machine learning can be effectively combined for music education in a way that is both practical and accessible.

While not intended to replace formal instruction, the Real-Time Conducting Tutor complements teacher-led learning by reinforcing foundational skills during independent practice. With further refinement and expansion, systems of this kind have the potential to play a meaningful role in modern conducting pedagogy.

9. References

[1] J. Khazaei, H. Rasoulzadeh, M. Hansen, and H. Al Osman. 2025. Continuous hand estimation approaches to quantifying hand tremor using computer vision: A survey. *arXiv preprint arXiv:2504.19460*.

[2] A. Foteinos, A. Therakomen, P. Raches, and A. A. Argyros. 2025. A survey on modern performance analysis tools for human movement. *arXiv preprint arXiv:2502.17143*.

[3] J. H. Fonteles and M. A. F. Rodrigues. 2021. User experience in a Kinect-based conducting system for visualization of musical structure. *Entertainment Computing*, 37, 100399. https://doi.org/10.1016/j.entcom.2020.100399

[4] C. Huang and R. Onate. 2025. Chorify is an intelligent desktop application to teach dance and correct motion using pose estimation and vibration band. *Computer Science & Information Technology (CS & IT)*, 29–39. https://doi.org/10.5121/csit.2025.152203

[5] J. Colby, F. C. Harris, C. D. Carthen, R. Kelley, C. Ruggieri, and S. M. Dascalu. 2018. MUSE: A music conducting recognition system. In *Proceedings of the International Conference on Information Technology – New Generations*, 363–369.

[6] S. Fahn, M.-K. Lien, and C.-H. Chang. 2019. A real-time music conducting gesture recognition system based on dynamic time warping. *Sensors*, 19, 2570. https://doi.org/10.3390/s19112570

[7] R. Choi and Y. Kim. n.d. BeatSaver: Conducting gestures to metronome beats. KAIST Project Report.

[8] Google MediaPipe. 2024. *MediaPipe Solutions*. https://ai.google.dev/edge/mediapipe/solutions

[9] A. Salgian, L. Burke, and J. Ernest. 2025. Visual analysis of conducting gestures. In *Proceedings of the International Computer Music Conference (ICMC 2025)*.

[10] B. Shneiderman, C. Plaisant, M. W. Cohen, N. Elmqvist, and N. Diakopoulos. 2016. *Designing the user interface: Strategies for effective human–computer interaction* (6th ed.). Pearson Education.

[11] Google. 2025. Gemini Model. https://deepmind.google/technologies/gemini/

[12] J. Robert. 2018. Pydub. GitHub. https://github.com/jiaaro/pydub

[13] Van Rossum, G. (2020). The Python Library Reference. Python Software Foundation. Retrieved from https://docs.python.org/3/library/tkinter.html

[14] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[15] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools. Retrieved from https://opencv.org/

[16] Shinners, P. (2000). Pygame. Retrieved from https://www.pygame.org/

[17] M. Abadi, P. Barham, J. Chen, et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org

[18] F. Chollet, et al. 2015. Keras. https://keras.io

[19] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. Neural Computation, 9(8), 1735–1780.
[20] C. Delong. 2023. VirtualConductor: Music-driven conducting video generation system. GitHub. https://github.com/ChenDelong1999/VirtualConductor
[21] S. Böck, F. Krebs, and G. Widmer. 2025. Beat This! Simplifying beat tracking with a folk music benchmark. GitHub. https://github.com/CPJKU/beat_this
[22] PyInstaller Development Team. 2024. PyInstaller. https://pyinstaller.org