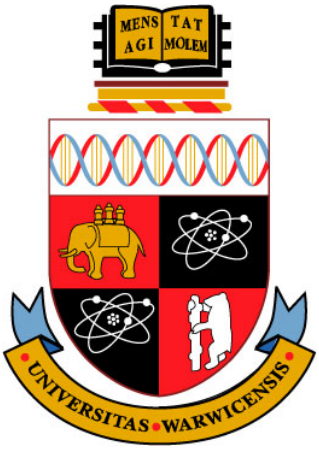

Creating a Coaching Aid System for Trampolining Routines

Author: Finlay Gray

Supervisor: Dr. Yu Guan

Year of Study: 3

Department of Computer Science

University of Warwick

April 2025

Abstract

This project presents an automated system leveraging computer vision and machine learning to detect, classify, and evaluate trampoline gymnastics routines. The objective is to provide an unbiased and consistent method of judging trampoline skills, enhancing both coaching feedback and athlete performance analysis. The system begins with advanced pose estimation using MediaPipe to accurately detect and track gymnast joint positions throughout routines. Subsequently, a temporal action localisation technique identifies bounce points, segmenting routine videos into individual trampoline skills. Each skill segment undergoes classification via a machine learning model specifically trained on trampoline skills. To objectively assess skill execution, a robust, rule-based deduction system was developed that aligned with established gymnastics scoring standards. This deduction model analyses detailed biomechanics data, including joint angles and positional accuracy, to ensure consistent scoring. The system also incorporates a PostgreSQL relational database, securely storing routine evaluations, historical performance records, and gymnast-coach interactions. A user-friendly interface, developed with PyQt5, provides coaches and gymnasts intuitive access to comprehensive performance analytics. It allows users to monitor skill progression, identify recurring execution errors, and generate personalised training insights. Ultimately, this integrated framework supports judges in delivering accurate, transparent scoring while empowering gymnasts and coaches with valuable analytical insights, significantly enhancing trampoline training and competition preparation.

Keywords: Computer Vision, Machine Learning, Software Development, Sports, Python, LSTM

Contents

1 Introduction

1.1 Overview

Trampoline gymnastics is a discipline where an athlete performs a routine consisting of ten skills on a trampoline. Invented in 1934 by George Nissen and Larry Griswold at the University of Iowa, trampoline was initially developed as a training tool for tumbling, another gymnastics discipline. It subsequently evolved into its own competitive sport, gaining significant international recognition when it was included in the Olympic Games in 2000.

Spectators watching trampoline events can witness athletes bouncing over eight meters into the air while executing intricate twists and somersaults (International Olympic Committee, 2024). Performances are judged based on four key criteria:

- **Execution**: Judges assess the precision and form of each element.
- **Difficulty**: Evaluates the complexity and risk of the routine.
- **Horizontal Displacement**: Measures how well the gymnast stays centered.
- **Time of Flight**: Calculates the total airborne time.

In this project, the focus is on execution judging. Traditionally, six execution judges score how well the skills are performed, deducting a maximum of 0.5 from each skill and then subtracting their total deductions from 10. The two median judges' scores are then added together to give an execution score out of 20.

Advancements in machine learning and computer vision have increasingly impacted sports analytics. In trampoline gymnastics and similar judged sports such as artistic gymnastics, diving, and figure skating, ML-driven systems have the potential to significantly reduce human error and bias. These technologies allow for precise analysis of athletes' movements, offering accuracy beyond traditional human judging capabilities.

1.2 Motivation

Despite having a detailed Code of Points set by the Fédération Internationale de Gymnastique (FIG), human judging remains inherently subjective. Scores awarded can vary between judges, and controversies or scoring inquiries are not uncommon in high-stakes events. This subjectivity and inconsistency have motivated interest in automated judging systems to assist or augment human judges, aiming for more objective and consistent evaluation.

This project presents the development of an ML-driven system for judging trampoline routines using computer vision, designed to complement human judges and serve as a training tool for athletes and coaches. The work addresses the challenges and ethical considerations of using ML in sports, such as limited training data, computational constraints, athlete privacy, and the broader role of ML in competitive settings.

2 Objectives

After analysis of the problem described above the following project objectives were created.

2.1 Must haves

1. Data Collection:
    - Set up a camera at Warwick Trampolining sessions to record gymnasts performing all necessary skills.
    - Ensure a wide range of variations of each skill by different gymnasts.
    - Collect and label additional data to improve model accuracy.
2. Skill Recognition & Judging:
    - Use a 3D pose estimation API to extract motion data from videos.
    - Recognise skills performed by gymnasts.
    - Implement a execution judging rule-based model.
    - Deduct execution scores based on comparison to the code of points.
    - Ensure execution scores for a 10-skill routine are within 0.5 of the benchmark score.
    - Enable the system to decompose a 10-skill routine into individual skills for model input.
3. User Interface & Experience:
    - Develop a functional GUI allowing users to insert videos and view the resulting score.
    - Ensure the GUI appears professional and includes analytical data visualisation.

2.2 Should haves

1. User Management & Experience:
    - Implement a login system for the application.
    - Support two types of users:
        - Gymnast users can upload videos, receive execution scores, and view personal analytics.
        - Coach users can manage gymnasts, view their analytics, and oversee their performance.
2. Persistent Data & Analytics:
    - Develop a database system to maintain persistent user data.
    - Allow gymnasts to track all past performances.
    - Enable coaches to analyse gymnast progress over time.

2.3 Could haves

1. Performance & Accuracy Enhancements:
    - Upgrade to multiple camera POVs to improve accuracy.
2. System Expansion & Features:
    - Augment the GUI to provide richer analytics.
    - Implement additional data-driven enhancements to refine execution judging.

2.4 Won't haves

1. Hardware-Based Enhancements:
    - Use sensors on trampolines for improved time of flight accuracy.

3 Literature Review

3.1 AI and Computer Vision in Other Sports

The use of AI for performance analysis and judging in sports has grown in recent years. Computer vision systems are widely used in sports such as Cricket (Hawk-Eye) and football (VAR) to routinely make calls and decisions that assist officials. However, these system mainly seem to focus on rule enforcement rather then qualitative scoring. In sports that require specific scoring, researchers have been exploring automated scoring systems that combine visual data with machine learning. For example, in competitive diving, (Okamoto, 2024) trained an AI model to score dives by analysing video of the takeoff, flight, and entry, aiming to predict judges' scores. This approach to diving evaluation combines neural networks (to extract features like diver pose and splash size) with rule-based analysis that mimics human scoring criteria. This hybrid method produced a more transparent and objective scoring system, indicating the potential of blending data-driven and expert-defined logic.

3.2 Gymnastics Judging Systems

Within gymnastics, the most notable development is the Fujitsu Judging Support System (JSS) co-developed with FIG (Fujitsu, 2023). This system has been created to judge up to 10 different artistic gymnastic apparatus. Initially, the JSS used wearable sensors, but it has since evolved into a purely vision-based 3D sensing system. High-speed, high-definition cameras capture the gymnast's performance from multiple angles, then pose estimation and 3D reconstruction algorithms model the gymnast's entire movement in fine detail. The system can automatically identify the elements performed and measure angles and positions to evaluate form. At the 2023 Gymnastics World Championships in Antwerp, the AI system was used on all apparatus as a supplementary tool for judges – for instance, to review inquiries about difficulty scores. Early reports indicate that it can precisely identify skills and provide consistent measurements, though final scoring decisions still rest with human judges. This real-world deployment underscores that AI judging is no longer just theoretical but becoming integral to elite competition.

3.3 Trampoline Skill Recognition Research

Specific to trampoline gymnastics, Connolly et al. (2017) conducted a study on automated skill identification from video. They utilised a single fixed video camera to record trampoline routines and applied open-source pose estimation, using the Stacked Hourglass CNN model for 2D human pose, to each frame. From the pose data, they extracted features like body orientation and joint angles over time, essentially capturing the trajectory of the gymnast's shape during each skill. These trajectories were then compared against a database of reference trajectories for known skills. A nearest-neighbour classifier with a mean squared error distance metric was used to match the performed skill to the closest reference skill. With a dataset of 714 examples covering 20 distinct trampoline skills, their system achieved an average skill recognition accuracy of 80.7%. Notably, this was accomplished without any special sensors or markers

on the athlete, relying purely on computer vision. Prior to this vision-based approach, trampoline skill classification had been attempted with wearable sensors; for instance, Helten et al. (2011)) got promising results using inertial measurement units (IMUs) attached to gymnasts to classify jump types. However, such suits are impractical in competition and against regulations, which makes vision-based methods far more attractive in practice.

3.4 Deductions from Existing Work

Connolly's work demonstrated the feasibility of identifying trampoline moves via pose estimation. It also highlighted some challenges: pose estimates from a single camera can be noisy or ambiguous (e.g., distinguishing twisting directions), and the approach focused only on skill identification rather than judging execution quality. The authors noted that extending to automated execution judging would be a logical next step. Execution judging is more complex because it requires assessing how well a skill is performed (body form, control, consistency) rather than just which skill it is. Some related work in rhythmic gymnastics by Díaz-Pereira et al. (2014) attempted automatic recognition and scoring of routines from video. Their method differed, but it similarly sought to quantify performance quality via computer vision. These studies, along with advances in human pose estimation (e.g. DeepPose, OpenPose, MediaPipe) and action recognition in videos, form the foundation for the trampoline judging system. This project aims to build upon these by not only recognising skills but also evaluating execution using a combination of learned models and rule-based criteria derived from the FIG Code of Points.

4 Project Management Approach

An Agile project management methodology was adopted to manage the development of the AI-based trampoline routine judging system. Agile emphasises iterative development and adaptability, breaking the work into smaller increments called sprints (Beck et al., 2001). In this project, a Scrum-style approach was used with 3-week sprint cycles. According to Rubin (2013), sprints typically last between two and four weeks, making a 3-week duration ideal for balancing rapid iteration with sufficient time to implement and test features. At the start of each sprint, specific goals and tasks were planned from the project backlog, and at the end, a review was conducted to evaluate progress and gather feedback. This iterative cycle enabled continuous improvement and flexibility, aligning with Agile values of "responding to change over following a plan" (Beck et al., 2001). For example, if a feature took longer than expected or user feedback suggested a change in priority, the plan for the next sprint was adjusted accordingly. Regular sprint reviews and retrospectives ensured that lessons learned were carried into subsequent sprints, and stakeholder input was incorporated frequently rather than only at the end of the project. As a result, the development could adapt to challenges (such as data quality issues or model accuracy concerns) early, rather than encountering them as surprises late in the timeline.

Task tracking was handled using Trello, an online Kanban-style board. All project tasks and user stories were initially listed in a backlog on the Trello board. Each task was represented as a card. During sprint planning, cards were moved from the backlog into the "In Progress" list. Upon completion, cards were shifted to the "Completed" list. This provided a clear visual overview of project status, consistent with standard Agile practices (Stellman and Greene, 2014).

Feedback integration was integral to the project. Fortnightly meetings were conducted with the projects supervisor, and in addition informal sprint review meetings were held with stakeholders, such as coaches or judges from the Warwick Trampolining Club. Feedback was documented and converted into actionable tasks on Trello. This ensured continuous stakeholder engagement, which is central to Agile project management (Cohen et al., 2004). For instance, after an early sprint demonstration, a user suggested that the user interface should display not just the final score but also a breakdown of execution deductions per skill. This feedback was logged and in the subsequent sprint, a task “Add score breakdown to UI” was implemented. In this way, the project leveraged the Agile principle of continuous stakeholder engagement, ensuring that the final product would meet user expectations.

In addition to the Agile sprint plan, a high-level Gantt chart was used to outline the project timeline and key milestones. While Agile is iterative, the academic project still had fixed deadlines (e.g. progress report, final submission), so the Gantt chart helped map out how the sprints aligned with these milestones. The chart plotted each 3-week sprint sequentially, with major deliverables for each. This provided an overview of the entire project schedule and ensured that critical paths were identified. For example, it was clear from the Gantt

timeline that data needed to be collected early to leave enough time for model training in later sprints. The Gantt chart also assisted in risk management; if one sprint's tasks were delayed, the chart made it easier to see which future tasks might need rescheduling. Figure 1 summarises the project timeline using a Gantt chart, showing the sequence of 9 sprints and their main focus areas. Each bar in the chart corresponds to a sprint, and the objective of each sprint is labelled below.

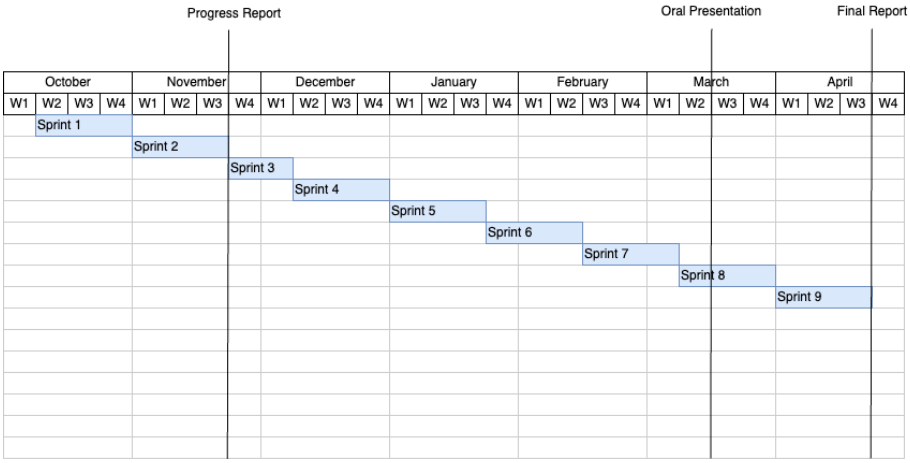

Figure 1: Gantt chart showing sprint timeline

Sprint 1: First round of data collection (small set of data) and implementation of 3D pose estimation API into software, creation of GUI.

Sprint 2: Second round of data collection, implementing and training ML model to perform skill recognition on small subset of skills.

Sprint 3: Data labelling.

Sprint 4: Upgrading ML model for skill recognition for full range of skills.

Sprint 5: Upgrade rule based execution scoring for full range of skills.

Sprint 6: Full execution scoring of each skill.

Sprint 7: Addition of analytical data as well as implementing login system and improved GUI.

Sprint 8: Finalising project, presentation.

Sprint 9: Final report.

Overall, the Agile project management approach (with structured sprints, ongoing feedback incorporation, and tools like Trello and Gantt charts) kept the project organized and adaptable. It allowed the project to be created and effectively executed within the given time frame, ensuring that by the end of the project the system met its requirements and was delivered on time.

5 Requirement Gathering

Defining clear requirements was crucial given the interdisciplinary nature of the project. The requirements were gathered from three primary sources:

- The author's own experience as a Great Britain (GB) acrobatic gymnast and qualified trampoline judge.
- Consultations with users at the Warwick Trampolining Club.
- Formal judging standards gathered from the FIG Code of Points (Fédération Internationale de Gymnastique, 2024) and competition rules.

By combining insights from personal domain expertise, end-user expectations, and official rules, a comprehensive set of functional and non-functional requirements was developed.

Firstly, the author's background in gymnastics and trampolining heavily informed the initial requirements. Having first-hand knowledge of how routines are judged in competitions, the author outlined what an automated judging system would need to do to be useful and accurate. Similarly, knowing common form faults and deductions, the author was able to show the system the specified major execution errors and deduct points accordingly. The author's experience also highlighted usability needs; as a gymnast, one would want feedback that is easy to interpret, and as a judge, one would want a tool that does not interfere with the flow of judging, meaning it should be reasonably fast and straightforward to use during or immediately after a routine. These considerations formed an initial requirements draft.

Secondly, consultation with end users provided another layer of requirement refinement. The project engaged members of the Warwick Trampolining Club, including fellow gymnasts, coaches, and judges, to gather their opinions on what features would be most valuable in an AI judging system. This was done through informal interviews and discussions. The feedback from these consultations was invaluable, as it brought out requirements that the author alone might not have considered. It ensured the system would address real needs and pain points of its intended audience. Notably, involving users at this early stage allows for good requirement engineering and contributes significantly to project success– their involvement helped validate that the project was on the right track feature-wise.

Thirdly, the formal judging criteria and rules provided by the FIG (Fédération Internationale de Gymnastique) Code of Points for Trampoline were used to derive explicit functional requirements related specifically to execution scoring. The Code of Points defines how execution is scored by averaging judges' deductions, including penalties for various infringements such as stepping off the trampoline, as well as other aspects like horizontal displacement and time of flight for elite competitions. Although the project's scope might not cover all advanced aspects, it needed at least to implement the standard execution scoring. Therefore, one requirement established was that "The system shall calculate the total execution score for a routine as per FIG Code of Points." This high-level requirement further breaks down into sub-requirements such as: "The system shall apply execution deductions for faults (legs apart, incorrect form,

etc.) consistent with FIG standards (e.g., small fault = 0.1, medium fault = 0.2, etc., per judge)." These criteria were directly taken from the judging handbook. Additionally, safety and format rules, like "a routine consists of 10 consecutive skills," led to requirements such as "The system shall analyse up to 10 skills per routine". By encoding these execution-specific domain rules as requirements, the project set clear targets for the software's functionality.

All identified requirements were documented in a formal requirements list. The list below presents a summarised subset of the key requirements. These requirements helped in project planning and validation – every implemented feature could be traced back to one or more of these requirements, and conversely, all listed requirements could be checked off as the project reached completion.

5.1 Full List of Requirements

5.1.1 Functional Requirements

FR1: Data Collection

- FR1.1: The system shall utilise video data collected from Warwick Trampolining sessions, featuring various gymnasts performing comprehensive trampoline skills.
- FR1.2: Collected data shall represent diverse trampoline skills and execution variations to ensure the generalisability of the ML model.

FR2: User Interface (UI)

- FR2.1: The system shall provide a graphical user interface (GUI) allowing users to upload trampoline routine videos.
- FR2.2: The GUI shall clearly display computed total scores for routines.
- FR2.3: The GUI shall present detailed feedback for each skill, including skill identification and execution deductions.

FR3: Pose Estimation

- FR3.1: The system shall incorporate a 3D pose estimation API to convert video footage into numerical pose data points.
- FR3.2: Pose estimation results shall be compatible with machine learning processing requirements.

FR4: Skill Recognition

- FR4.1: The system shall recognise trampoline skills performed by gymnasts from video data.
- FR4.2: Skill recognition accuracy shall be at least 80% for standard trampoline routines.

FR5: Temporal Action Localisation

- FR5.1: The system shall accurately determine the start and end times of each identified skill within the trampoline routine video.

- FR5.2: Temporal localisation accuracy shall be sufficient to ensure precise execution scoring aligned with standard trampoline judging practices.

FR6: Execution Scoring

- FR6.1: The system shall calculate execution scores based on detected faults according to the FIG Code of Points.
- FR6.2: Execution scores calculated by the system shall be accurate within ± 0.5 points of scores provided by qualified human judges for standard 10-skill routines.

5.1.2 Non-Functional Requirements

NFR1: Performance and Efficiency

- NFR1.1: The system shall process and return results for a standard 10-skill trampoline routine video within one minute.

NFR2: Usability

- NFR2.1: The UI shall be intuitive, requiring minimal training.
- NFR2.2: Results and scores provided by the system shall be easily interpretable by gymnasts, coaches, and judges.

NFR3: Accuracy and Reliability

- NFR3.1: Overall scoring accuracy shall consistently match professional human judges within ± 0.5 points.
- NFR3.2: Skill recognition shall be robust against typical performance variations, camera angles, and lighting conditions.

NFR4: Maintainability and Extensibility

- NFR4.1: System architecture shall support updates to skill databases, scoring rules, and additional feature integrations.

NFR5: Scalability

- NFR5.1: The database and application structure shall support scaling for additional users and routines without performance degradation.

Every requirement shown above was later used as a reference point during development and testing. For example, to verify the "Skill Identification" requirement, a test was done using sample videos to ensure all 10 skills in a routine were indeed captured by the system. In cases where discrepancies arose, it indicated either a bug or a misinterpretation of the code of points, which was then corrected. This requirements-driven validation ensured that by the end of the project, the system met the needs and expectations initially set out.

6 Legal, Social, Ethical and Professional Issues & Considerations

Given the requirement to collect training data involving human participants—specifically students from Warwick University—careful consideration was necessary to address legal, social, ethical, and professional responsibilities throughout the research process. This section outlines key considerations and actions undertaken to ensure compliance and uphold high ethical standards.

6.1 Ethical Approval

To ethically conduct the data collection necessary for training the trampoline judging system, an "Ethics Application Form for Student Research Project" was completed and submitted for review by Warwick University's Ethics Committee. This process involved a thorough examination of the proposed methodologies, ensuring all research activities would align with established ethical standards protecting participant welfare, confidentiality, and rights.

Ethical approval was granted conditionally upon the provision of detailed documentation to inform and protect participants. This additional documentation included a comprehensive Participant Information Leaflet (PIL) and a clear, explicit written consent form. These documents informed potential participants about the study's purpose, procedures, potential risks, and benefits, as well as clarifying their rights to withdraw at any time without consequence.

6.2 Participant Information Leaflet (PIL)

The Participant Information Leaflet provided clear and transparent information to participants regarding:

- The project's purpose and objectives.
- Procedures for data collection, including details of video recording and how the collected footage would be utilised.
- Potential risks and benefits of participation.
- Assurance of participant anonymity and data confidentiality.
- Participant rights, particularly the voluntary nature of involvement and the right to withdraw from the study at any point without penalty.

6.3 Informed Consent Form

The written consent form served as formal documentation to ensure explicit participant consent, capturing participants' acknowledgment and understanding of:

- Their voluntary participation and understanding that their involvement could be terminated at any stage without penalty.

- How data would be used, stored securely, and disposed of after project completion.
- Measures undertaken to protect their privacy and confidentiality.

Participants were required to sign this consent form before any data collection commenced, ensuring clear mutual understanding and agreement.

6.4 Data Privacy and Confidentiality

All data collected during the project was stored securely on password-protected, encrypted devices with restricted access. Measures were taken to anonymise participants, ensuring that personal identifiers were removed or obscured in any published or shared materials, maintaining strict compliance with data protection regulations.

7 System Design

The design of the system utilises a modular approach, separating out each main stage in the process. This facilitates maintainability and allows for future expansion, as each component can be developed and tested independently before integration.

7.1 System Architecture Overview

The system follows a pipeline architecture in which raw video data of a trampoline routine passes through several preprocessing stages to yeild a final score. Figure 2 illustrates the high level architecture.

The key components include:

- A video capture and preprocessing module
- A pose estimation module
- A skill segmentation and feature extraction module
- A machine learning skill classification module
- An execution scoring module
- A results integration module

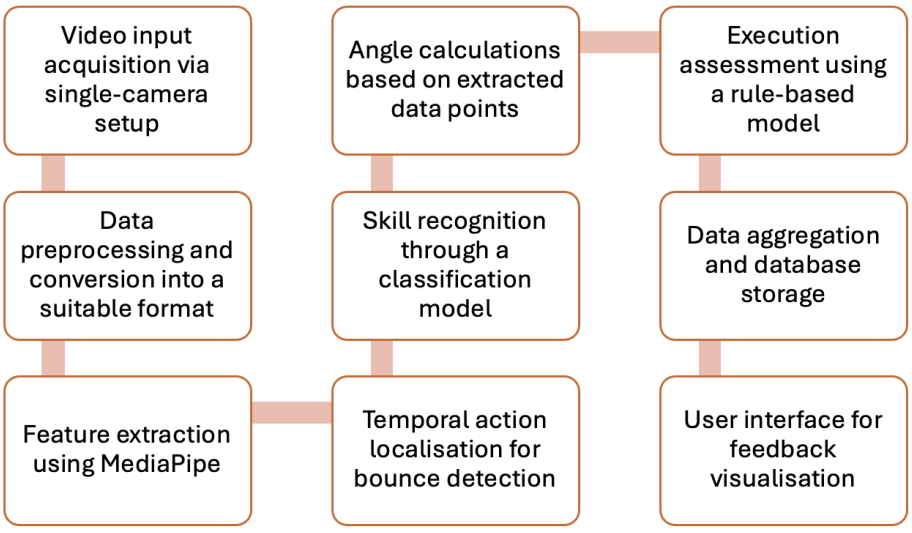

Figure 2: High-level flowchart showing system architecture

The video capture and preprocessing module takes in the input video as a recorded video file. The video file is then colour corrected using the Python CV2 library and passed into the pose estimation module. This module uses a computer vision model to detect the gymnast's body pose in each frame, producing a set of coordinates for key body landmarks. The project employs Google's

MediaPipe Pose solution, which provides high-fidelity tracking of 33 anatomical key points in real time. MediaPipe's framework estimates the 3D location of landmarks from frames, leveraging a lightweight model that can run at real-time speeds on standard hardware. This approach is markerless (no physical sensors on the athlete), aligning with the need for a non-intrusive judging aid.

After pose estimation, the data flows into the skill segmentation and feature extraction stage. A trampoline routine consists of a sequence of up to ten skills which must be identified individually for scoring. Segmentation involves detecting the start and end of each skill within the sequence. In trampoline gymnastics, a natural way to segment skills is by detecting the moments of contact with the trampoline bed – each skill corresponds to the aerial phase between take-off and landing on the bed. By tracking the athlete's vertical motion, the system can partition the pose time series into skill-specific segments. For example, the vertical position of the athlete's centre of mass typically exhibits peaks (highest point in a jump) and troughs (landing compression on the bed). These patterns allow identification of when one skill ends and the next begins. The segmentation process also uses temporal smoothing to avoid false splits due to minor bounces.

Once segmented, each skill's pose data is processed to extract descriptive features. In the design, two forms of features are considered: raw joint coordinates over time, and kinematic features derived from those coordinates (such as angles between limb segments, body orientation, etc.). Previous research by Connolly demonstrated the effectiveness of using pose-based features (joint angles and their trajectories) for trampoline skill recognition. The system builds on this insight: for each skill segment, the system computes time series of joint angles (e.g., knee angle, hip angle, etc.) and orientations which capture the shape and motion of the athlete during that skill.

The skill classification module is the core AI component that identifies what skill was performed in each segmented portion of the routine. I formulated this as a multi-class classification problem, where the classes correspond to possible trampoline skills (e.g., tuck jump, pike jump, back somersault, etc.). The set of skill classes in the system's scope is defined based on common competition elements and the available training data. During system design, consideration was given to the choice of model. Given that skill classification from pose time-series is essentially an action recognition task, models that can handle sequential data were prioritised. I selected a deep learning approach using a Recurrent Neural Network (RNN) architecture, more specifically a Long Short-Term Memory (LSTM) network, to leverage temporal patterns in the pose data. LSTM networks are well-suited to sequence data because they maintain an internal state that can capture information from earlier in the sequence when making predictions about later points. The architecture takes the sequence of pose features for a skill as input and outputs a probability distribution over the skill categories.

The execution scoring module, which is described in detail in Chapter 9, takes the pose sequences and the identified skill labels to evaluate how well each skill was performed. This module applies rule-based analysis inspired by the FIG

Code of Points guidelines for execution deductions. Essentially, it examines specific technical aspects of the performance (body form, consistency, control) and assigns deductions for any deviations from perfect execution. The design decision to use a rule-based system, as opposed to a learned model, for execution was made for two reasons: first, the criteria for execution deductions are explicitly defined by gymnastics regulations, making a rules-based approach natural; second, acquiring a sufficiently large and labelled dataset of routines with execution scores to train a supervised model was not feasible within the project scope.

Finally, the system includes a results integration and user interface component. Once execution scores are determined for a routine, they are combined into a final score in the same manner as official competitions. The results, along with details such as identified skills and deduction reasoning, are then stored in a database and presented to the user via an application interface. The user interface enables gymnasts and coaches to upload videos and receive feedback. The system supports two categories of users: gymnasts and coaches, each with different interface views and data access. The database holds persistent data including user accounts, uploaded videos metadata, skill analysis results, and historical performance for analytics.

8 Model Development and Training

A core component of the system is the machine learning model for skill classification. Developing this model required several steps: data collection and preprocessing, feature extraction, designing the neural network architecture, training the model with appropriate techniques (including cross-validation), and evaluating its performance. This chapter provides a detailed account of these steps, highlighting the technical depth of the model development process.

8.1 Data Collection and Preprocessing

The first challenge was assembling a dataset of trampoline performances with labels indicating the skill performed. Given the project's scope and ethical considerations, the dataset was relatively limited. It comprised video recordings from training sessions from Warwick University Trampolining Society, supplemented by a small number of publicly available trampoline routine videos. Each video was typically a full routine containing up to ten skills. For training the classifier, these routines needed to be segmented and labelled per skill. The videos were manually annotated: the start and end frame of each skill were marked, and each segment was given a label from a predefined set of skills (e.g., "Tuck Jump", "Straddle Jump"). In total, the curated dataset included 650 individual skills, covering 10 different skill types. These skills are listed below:

- Tuck jump
- Straddle jump
- Pike jump
- Seat drop
- Seat to feet
- Half to seat
- Half to feet
- Half turn
- Full turn
- Tuck back

This set of skills was chosen to strike a balance between model accuracy and training data availability, as well as providing enough skills to allow for a large range of unique routines to be performed and judged. After obtaining raw video segments for each skill, the system applied pose estimation to convert them into a form suitable for machine learning. MediaPipe's Pose model was utilised to extract 3D landmarks for 33 keypoints on the gymnast's body for every video frame. Each key point comes with (x, y, z) coordinates. The full list of key points given is shown in Figure 4. In Figure 3 below we can see an example of a screenshot showing how MediaPipe tracks body landmarks. In each video frame, the pose is represented by a vector of length 99 (33 points * 3 coordinates).

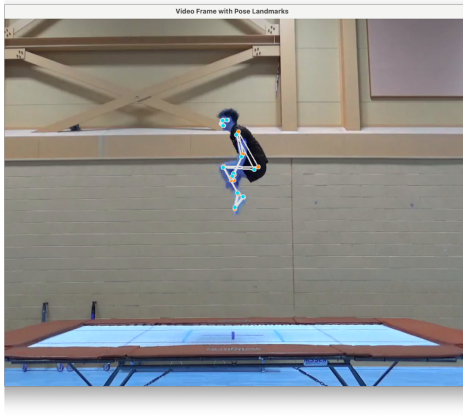

Figure 3: MediaPipe tracking example

- 0 - nose
- 1 - left eye (inner)
- 2 - left eye
- 3 - left eye (outer)
- 4 - right eye (inner)
- 5 - right eye
- 6 - right eye (outer)
- 7 - left ear
- 8 - right ear
- 9 - mouth (left)
- 10 - mouth (right)
- 11 - left shoulder
- 12 - right shoulder
- 13 - left elbow
- 14 - right elbow
- 15 - left wrist
- 16 - right wrist
- 17 - left pinky
- 18 - right pinky
- 19 - left index
- 20 - right index
- 21 - left thumb
- 22 - right thumb
- 23 - left hip
- 24 - right hip
- 25 - left knee
- 26 - right knee
- 27 - left ankle
- 28 - right ankle
- 29 - left heel
- 30 - right heel
- 31 - left foot index
- 32 - right foot index

Figure 4: 33 key points given by MediaPipe

However, using raw coordinates directly has downsides as they depend on the camera angle and the athlete's position in the frame. To address this, normali-

sation and augmentation was performed as shown below:

- Orientation Normalisation: Because flips and twists in trampoline can be performed in different orientations relative to the camera (e.g., a back somersault could be performed facing to either the left or the right of the camera), I considered normalising or augmenting orientations. The approach used to augment the data is called mirroring; I horizontally flipped each video to create a mirrored version. This effectively doubles the dataset and makes the model invariant to left-right orientation of the skill. For example, a half-turn jump performed facing left vs right should be classified the same, and mirroring helps the model learn that symmetry.
- Temporal Padding: Each skill segment can vary in length (duration). Directly feeding sequences of varying length into a fixed-architecture neural network is not feasible without padding. I chose a fixed length T (in frames) which was the length of the longest input. Shorter sequences were padded with a special padding value to reach length T. This zero-padding approach is standard and the neural network was designed to ignore the padded time steps (as described later).
- Label Encoding: Skill labels (which were names of skills) were encoded using one-hot encoding for training the classifier. For example, with 20 skill classes, a "Tuck Jump" might be encoded as [0,0,...,1,...0] (with a 1 in the index corresponding to that class).

8.2 Model Architecture

The final architecture of the skill classification model can be summarised as follows and can be viewed in Figure 5:

- **Input layer:** A sequence of length T with feature dimension D. The feature dimension D corresponds to the number of pose features per frame and so D = 99. The length T reprents the longest value of all inputed training data. The input thus is a $T \times D$ matrix representing the time-series of a single skill.
- **LSTM layers:** The first hidden layer is an LSTM with 128 units. This processes the input sequence and produces an output at each time step. I added a second LSTM layer with 64 units to further abstract temporal patterns. Stacking LSTMs can capture higher-level temporal features, as the first LSTM might capture short-term motion patterns and then the second might capture the overall sequence trend. I applied a dropout of 0.2 between LSTM layers as regularisation to reduce overfitting given the limited data.
- **Output layer:** A dense layer with size equal to the number of skill classes, using a softmax activation function. The softmax layer produces a probability distribution over the classes for the input sequence. The predicted class is the one with highest probability.

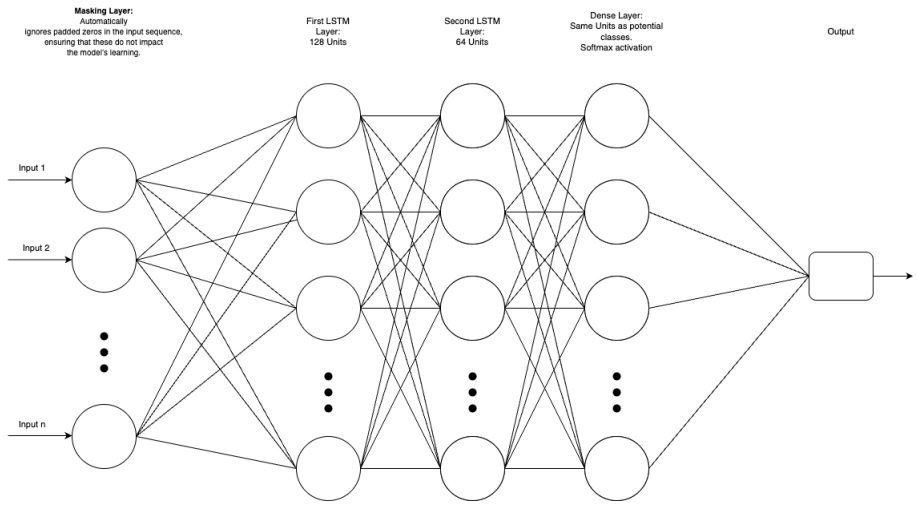

Figure 5: MLP architecture

The model was implemented using TensorFlow and Keras, which provided high-level APIs to quickly iterate on architectures. I also evaluated alternative algorithms for classification to check I had the most efficient algorithm, such as a classical k-Nearest Neighbors on the angle trajectory features, inspired by Connolly et al. (2017). Those simpler models yielded lower accuracy, which reinforced the choice to use a trainable deep network that could better handle the complexity of the data.

8.3 Training, Validation and Hyper-parameter tuning

In order to ensure a fair and unbiased estimate of model performance, the available dataset was partitioned into two distinct subsets prior to any training or hyper-parameter tuning: a training set and a held-out test set. The test set remained completely unseen during the model development and hyper-parameter tuning stages, thereby preventing any information leakage.

8.3.1 Model Training and Cross-validation:

The model was trained using a supervised learning approach with a categorical cross-entropy loss function, which is appropriate for multi-class classification with a softmax output. The optimisation was performed using the Adam optimiser at a learning rate of 0.001, a common choice to promote quick and stable convergence. Given the limited amount of data available, a 5-fold cross-validation strategy was employed on the training set. In each of the 5 iterations, the training data was shuffled and split into 5 folds – with one fold held out as the temporary validation set while the remaining 4 folds served as the training data. The performance metrics were then aggregated across folds to assess the model's generalisation capability and variability. Consistent cross validation accuracies across folds provided confidence that the model was robust and not overly sensitive to specific training samples.

For each cross validation run, training was performed in mini-batches of 16 sequences. Each epoch involved iterating over all training sequences once, with an initial cap of 30 epochs. In many instances, the model converged in under 20 epochs, and early stopping was introduced (monitoring the validation loss and halting training after 5 consecutive epochs without improvement) to mitigate overfitting.

To further address class imbalance (where certain skill classes were under-represented), class weights were computed inversely proportional to class frequencies and incorporated into the training process. This adjustment improved the recall for less common classes.

8.3.2 Hyper-parameter Tuning:

Hyper-parameter tuning was conducted manually within the cross validation framework on the training data to select the optimal configuration. The parameters explored included:

- **Number of LSTM units** - experiments were conducted with 64, 128, and 256 units. It was observed that using more than 128 units offered diminishing performance benefits and increased the risk of overfitting.
- **Number of LSTM layers** - both 1 and 2 layers were evaluated, with 2 layers providing slight improvements.
- **Learning rate** - comparisons between 0.001 and 0.0005 indicated that 0.001 achieved faster and more consistent convergence.
- **Batch size** - batch sizes of 16 and 32 were tested, with 16 proving more effective given the small dataset size.

During each fold, the model checkpoint with the lowest validation loss was saved. The average performance across folds guided the selection of the optimal hyper-parameters.

8.3.3 Final Model Training and Testing:

After determining the optimal hyper-parameters using k-fold cross validation exclusively on the training data, a final model was re-trained on the entire training set using the same number of epochs as observed for the best-performing fold. This final model was then evaluated on the unseen test set. Evaluating on the test set – which was set aside at the beginning of the process – ensured that the performance metrics reflect the model's true generalisation ability without any risk of information leakage.

9 Execution Scoring and Rule-Based Adjustments

While the skill classification model provides the means to assess what skill was performed, an equally important aspect of the judging system is assessing how well the skill was executed. In trampoline gymnastics, execution scoring involves detecting imperfections in form and technique and deducting points accordingly. Execution starts from a maximum and deductions are subtracted for each fault observed. This chapter details the development of a rule-based execution scoring system that uses the pose data and skill information to evaluate performance quality. The system is rooted in the FIG Code of Points rules and tries to emulate a human judge's decision-making by algorithmically calculating angles and deviations in the gymnast's form.

9.1 FIG Code of Points and Execution Criteria

The FIG Code of Points for Trampoline outlines various criteria for execution deductions. Execution judges watch for faults such as:

- Bent knees: Not maintaining straight body lines when they should be straight.
- Leg separation: Legs should be kept together unless the skill requires them apart (like straddle jumps); unintended separation incurs deductions.
- Incorrect body position in twists or flips
- Other faults: Such as not landing on both feet simultaneously, brushing the bed with hands or knees, etc. (some of these are larger penalties or even non-standard deductions).

The Code of Points assigns specific deduction values (often in increments of 0.1) for these faults. For instance, a slight knee bend might be a 0.1 deduction and a deeper bend 0.2. Similarly, failure to keep feet together on landing might be 0.1–0.2 depending on severity.

9.2 Rule-Based Deduction System Design

The execution scoring was defined as a set of rules, each corresponding to one type of potential deduction. Each rule checks the pose data and skill type to determine if a deduction should be applied for a given skill performance. The rules are applied to each skill segment of the routine, and the deductions are summed up. The main rules implemented can be seen in detail in Table 1 below.

Each rule yields a deduction value (which could be 0 if no fault, or 0.1, 0.2, etc., for a fault). These deductions are summed for each skill. The system effectively simulates a single "composite" execution score. It was chosen to scale the execution score in a simplified manner: we assume a base of 10.0 (as if one judge) and then double it to scale to 20.0 (as if two judges), to roughly simulate the typical range. Specifically, the formula the system used for the final E-score of the routine was:

$$E_{\text{final}} = (10 - \sum \text{deductions}) \times 2.$$

Jump Type	**Angles Evaluated**	**Detailed Thresholds**	**Common Deduction Reasons**	**Evaluation Frame Window**
Straddle/Pike Jump	Leg vertical angles, Knee angles, Pointed toes angles	Leg angle $\geq 145°$: 0.0 Leg angle 90°-145°: 0.1-0.2 Leg angle 65°-90°: 0.2-0.5 Leg angle $< 65°$: 0.5 Knee angle $\leq 120°$: 0.1 Toes angle $< 30°$: 0.1	Loose Shape, Legs Bent, Toes Flexed	Middle 20% (40–60%)
Tuck Jump	Knee vertical angles, Pointed toes angles	Knee angle $\geq 135°$: 0.0 Knee angle 90°-135°: 0.1 Knee angle $< 90°$: 0.2 Toes angle $< 30°$: 0.1	Loose Shape, Toes Flexed	Middle 40% (30–70%)
Tuck Back	Shoulder-Hip-Knee angles, Pointed toes angles	Shoulder-Hip-Knee $\leq 45°$: 0.0 Shoulder-Hip-Knee 45°-90°: 0.1 Shoulder-Hip-Knee $\geq 90°$: 0.2 Toes angle $< 30°$: 0.1	Loose Shape, Toes Flexed	Middle 40% (30–70%)
Seat Drop Variations (seat_drop, seat_to_feet, half_to_seat, half_to_feet)	Knee angles, Legs together, Pointed toes angles	Legs apart: 0.1 Knee angle $\leq 140°$: 0.1 Toes angle $< 30°$: 0.1	Legs Apart, Legs Bent, Toes Flexed	Middle 60% (20–80%)
Twist Variations (half_turn, full_turn)	Knee angles, Legs together, Pointed toes angles, Trunk angle	Legs apart: 0.1 Knee angle $\leq 140°$: 0.1 Toes angle $< 30°$: 0.1 Trunk angle $< 165°$ or $> 195°$: 0.1	Legs Apart, Legs Bent, Toes Flexed, Off Balance	Middle 60% (20–80%)

Table 1: Detailed Execution Judging Rules and Angles Evaluated for Trampoline Routine Assessment (Rotated)

This way, if a routine had no deductions at all, $\sum \text{deductions} = 0$ and $E_{\text{final}} = 20$. If, say, a total of 1.5 in deductions were accumulated, $E_{\text{final}} = (10-1.5)\times 2 = 17$.

The execution rules were initially tested on a small set of example performances that the author and a coach qualitatively scored. For instance, a video was taken of a gymnast performing a routine decently but with noticeable form breaks (a slightly cowboyed [legs apart] somersault, bent legs in a straddle jump). The rule system was run and produced deductions: e.g., "Skill 3 (Back Somersault): legs apart 0.2. Summing these gave an execution score that roughly matched what the coach estimated (perhaps the coach said "around 8.0 out of 10 execution" which would be 16/20; the system might have given 15.8). This informal validation was encouraging.

The execution module was also made somewhat configurable: the threshold angles and deduction values were kept as constants that could be tweaked. This was useful when refining the system. For example, initial tests showed the toe point deduction triggered too often due to noise so then the threshold was raised to only catch obvious misses of toe point.

Another advantage to using the rule-based system is it allows the users to view the exact reason for each deduction given. This benefits coaches and gymnasts as it allows to not just to see their score, but also offers something they can improve on.

9.3 Limitations of the Rule-Based Approach

It is important to acknowledge the limitations of the current execution scoring system. Being rule-based, it can only detect what it has been explicitly programmed to detect:

- It may miss some qualitative aspects judges look for (like overall aesthetics, precise timing of movements).
- Its accuracy heavily depends on the accuracy of pose estimation. If the pose keypoints are off (e.g., losing track of ankles when they blur), the angle calculations can be erroneous. However, I did implement smoothing of the pose data to mitigate jitter (a moving average filter over a few frames to smooth out sudden unrealistic changes).
- Threshold tuning was based on limited observation and some borrowed values from the Code of Points. In reality, judges use their experience to gauge severity; our fixed numeric thresholds might not always correspond perfectly to human perception of "slight" vs "medium" error. For example, a knee angle of 160° might be a medium deduction in one context but if it was just momentary maybe judges consider it minor. The system doesn't understand context length of fault.
- The combination of deductions: Human judges have a sense of not "double-deducting" too harshly for essentially the same error manifesting in multiple ways. The system might independently deduct for knee bend and for

body alignment, even though they stem from the same root cause (knees bent cause body not straight). This could lead to slightly harsher scoring than a judge might give in total.

- A machine learning approach was not incorporated to execution due to data constraints, but such an approach could learn complex patterns of errors. The hybrid model (ML + rules) was a stretch goal; in the end, the system delivered the rule system alone. The implication is that the system might not correlate perfectly with real execution scores in competition, but it provides a reasonable proxy.

Despite these limitations, the rule-based execution scoring provided a functioning module that contributed to the overall goal: giving gymnasts automated feedback. Users testing the system found the breakdown of deductions understandable and generally in line with their own impressions of the performance.

In summary, the execution scoring component uses explicitly coded rules to translate pose-derived measurements into point deductions, striving to emulate the FIG judging criteria. While not perfect, it adds an essential dimension to the AI judging system: not just identifying what skills are done, but also critiquing how well they are done. Together with the classification model, it forms a comprehensive judging solution.

10 Temporal Action Localisation

To accurately segment trampoline routines into individual bounce events, a custom temporal action localisation algorithm was implemented. This approach was designed to identify discrete bounce frames within continuous motion using pose landmarks, velocity estimation, and motion thresholds. The following outlines the complete methodology implemented in the temporal bounce separation function:

10.1 Overview

The goal of the temporal segmentation function was to isolate each bounce from a full routine video by analysing changes in dynamic center velocity and hip height. The process is composed of several computational stages.

10.2 Video Feature Processing

Pose Landmark Extraction: The function begins by converting the input video, which extracts frame-wise skeletal landmark data. This includes 3D coordinates of joints such as shoulders and hips.

Torso Calibration: For each frame, the midpoint of both shoulders and both hips is calculated, and the average vertical difference between these is computed to define a "baseline torso length". This length acts as a reference to detect meaningful changes in posture during motion.

10.2.1 Hip Position Thresholding

Dynamic Hip Tracking: For each frame, the average vertical position of the hips is computed and stored. The 95th percentile of all hip positions is determined to approximate the typical landing level, which defines the bounce threshold. A dynamic threshold is set at 80% of this maximum to help filter out false positives caused by subtle movement or noise.

10.2.2 Dynamic Center and Velocity Estimation

Dynamic Center Calculation: The dynamic center is computed per frame using a weighted average of keypoint motion magnitudes. The center, hip position, and torso length are stored per frame for further evaluation.

Velocity Smoothing: A weighted velocity signal is computed by evaluating the difference in dynamic centers across a window of frames (`delta=3`). A moving average filter is applied to smooth the velocity signal, and outliers are clipped using a fixed velocity threshold of 1.0 to ensure clean peak detection.

10.3 Bounce Detection Algorithm

Zero-Crossing Detection: The algorithm identifies bounce frames as local zero-crossings in the smoothed velocity signal, where the weighted velocity changes from positive to non-positive.

Estimated Bounce Frame: To improve frame-level precision, the algorithm interpolates the bounce position by calculating a sub-frame correction based on the gradient of the velocity curve at the zero-crossing.

Hip Position Validation: For each detected bounce, the average hip position at the candidate frame is compared to the earlier-defined threshold. Bounces are only accepted if the hip position is within the threshold value.

Minimum Separation Rule: To avoid misclassification of rapid movements as separate bounces, a minimum bounce gap of 1 second is enforced between consecutive bounce detections.

10.4 Segment Creation and Visualisation

Bounce Segmentation: Frame indices corresponding to detected bounce events are used to slice the original video into segments, each representing an individual bounce. The first and last segments capture the video before the first bounce and after the last.

User Feedback and Playback: During testing, each bounce segment is displayed with annotated diagnostics, including the reasoning for bounce identification (momentum value, hip height). This step allows for visual inspection and interactive confirmation of bounce localisation accuracy.

10.5 Summary

This implementation of temporal action localisation provides a robust framework that combines rule-based motion analysis, dynamic positional thresholds, and advanced signal processing techniques to achieve accurate segmentation of trampoline bounce routines. This accurate segmentation is crucial, as incorrect segmentation would compromise the entire model's ability to produce reliable and precise outputs. By initially extracting pose landmarks and performing torso calibration, the method establishes a consistent anatomical reference that supports precise motion tracking. Subsequent dynamic hip tracking and thresholding effectively differentiate actual bounce events from minor movements and noise, enhancing the reliability of detections.

The integration of dynamic center calculations and velocity smoothing techniques significantly improves the algorithm's capability to discern meaningful movement patterns. The smoothed velocity signal ensures that subtle yet critical changes in motion direction, indicative of bounce events, are accurately captured through zero-crossing detection. Furthermore, interpolation at detected zero-crossings provides sub-frame precision, elevating temporal accuracy to a highly refined level.

The addition of stringent validation through hip position checks and enforced minimum separation intervals between detected bounces further solidifies the robustness of the algorithm, preventing false positives and closely spaced movements from compromising the integrity of bounce segmentation.

Finally, the segmentation and visualisation components enable practical usability, allowing for interactive review and verification of bounce detection through annotated video segments. This comprehensive methodology results in precise, temporally isolated video clips for each bounce event, significantly facilitating subsequent analyses, such as performance assessment and biomechanical evaluation. Overall, the hybridised analytical approach ensures consistent and high-quality segmentation across diverse trampoline routines.

11 Database and Application Development

With the core analytical components (skill classifier and execution scorer) in place, the focus shifts to the development of the database and application that surrounds these components. The database is essential for persisting data such as user profiles, uploaded videos, and results of analyses, while the application provides the user-facing interface to interact with the system. This chapter describes the design of the database schema, the implementation of the application for gymnasts and coaches, and how everything is connected. The aim was to create a functional prototype that demonstrates how the AI judging system can be used in practice by end-users.

11.1 Database schema and implementation

The database used for this project is a PostgreSQL relational database. The schema was designed to accommodate the following data entities:

- Users: Storing user account information.
- Videos: Each uploaded performance video is an entity with fields like upload date, the user who uploaded (or the coach and the gymnast it belongs to), and a reference to the stored video file path.
- Results: After processing a video, the results, scores and details, are stored.

The rationale for this design is to support multiple users and many-to-many relationships (one coach to many gymnasts, each gymnast with many videos, etc.). Figure 6 depicts the entity-relationship diagram of the database.

The database contains six tables:

1. submissions: This table links each video submission to the account that submitted it, also storing the final execution score and the data for the submission.
2. users: This table stores each users id, username, password and which type of account they possess.
3. coach_gymnast: This table links coaches to gymnasts.
4. skill_scores: This table links each separate skill performed to the submission it belongs to, as well as the skill name and the deduction value given to the skill.
5. deduction_details: This table links each skill with details of why each deduction it was made.
6. group_invites: This table stores invitation requests from a coach to add a gymnast to their group.

For this project, the PostgreSQL database is hosted locally using a Docker container. This approach provides a consistent and isolated development environment, eliminating the need for a full PostgreSQL installation on the host machine. It also simplifies deployment and ensures the setup can be easily replicated across different systems.

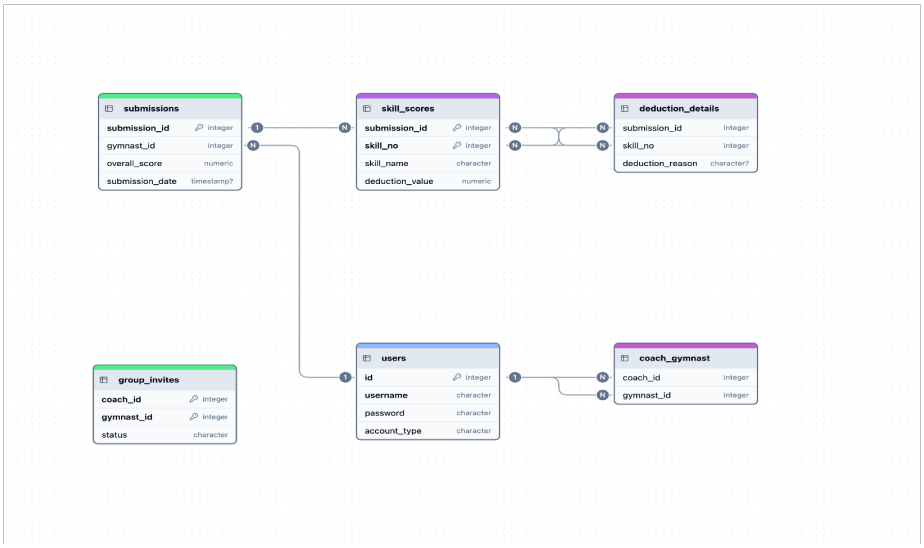

Figure 6: Database schema for the trampoline judging application. Key entities include Users (coaches and gymnasts), Videos (uploaded routines), and Results of analysis. Relations connect coaches to gymnasts and gymnasts to their performances and results.

The database container is started using the following command:

```
docker run --name my_postgres \
  -e POSTGRES_DB=trampolining \
  -e POSTGRES_USER=postgres \
  -e POSTGRES_PASSWORD=postgres \
  -p 15432:5432 \
  -d postgres:latest
```

Advantages of Using Docker for Local Database Hosting:

- **Environment Isolation:** The database runs inside a containerised environment, ensuring it does not interfere with or depend on other software installed on the host machine. This isolation reduces the risk of conflicts and makes debugging more straightforward.
- **Consistency Across Systems:** Docker ensures that every developer or automated system using the project runs the database in an identical environment, eliminating portability problems.
- **Ease of Setup and Deployment:** Developers can spin up a fully functional database with a single command, streamlining onboarding and minimizing time spent configuring infrastructure.
- **No Native Installation Required:** There's no need to install PostgreSQL directly on the host machine, which can be particularly beneficial for maintaining a clean development system or working on multiple projects with differing database requirements.

- **Portability and Version Control:** The Docker configuration can be included in the project's version control system, enabling full reproducibility and easier collaboration within teams.

This setup is particularly valuable in modern software development workflows, where reproducibility, ease of use, and scalability are key priorities.

11.2 Application architecture and user interaction

The user interface for this project was developed as a desktop application leveraging PyQt5, a robust and versatile Python binding for the Qt application framework. PyQt5 was selected due to its ease of use, extensive feature set, cross-platform compatibility, and strong integration with Python, allowing seamless integration with the project's backend Python code.

11.2.1 Overall Structure

The application follows a structured, multi-page design managed through distinct QWidget classes. Navigation between pages is smoothly handled via a central `QStackedWidget`, ensuring intuitive user experiences. The code setting up the structure of this can be viewed below.

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Trampolining Judging System")
        self.setGeometry(200, 200, 800, 600)
        self.current_user = None
        self.stacked_widget = QStackedWidget()
        self.setCentralWidget(self.stacked_widget)
        self.login_page = LoginPage(self)
        self.register_page = RegisterPage(self)
        self.home_page = HomePage(self)
        self.upload_page = UploadPage(self)
        self.display_score_page = DisplayScorePage(self)
        self.data_page = DataPage(self)
        self.profile_page = ProfilePage(self)
        self.gymnast_search_page = GymnastSearchPage(self)
        self.invitations_page = InvitationsPage(self)
        self.stacked_widget.addWidget(self.login_page)
        self.stacked_widget.addWidget(self.register_page)
        self.stacked_widget.addWidget(self.home_page)
        self.stacked_widget.addWidget(self.upload_page)
        self.stacked_widget.addWidget(self.display_score_page)
        self.stacked_widget.addWidget(self.data_page)
        self.stacked_widget.addWidget(self.profile_page)
        self.stacked_widget.addWidget(self.gymnast_search_page)
        self.stacked_widget.addWidget(self.invitations_page)
        self.create_menu()
        self.stacked_widget.setCurrentWidget(self.login_page)
```

11.2.2 Interface Pages and Functionalities

LoginPage Allows users to securely log in by providing credentials, ensuring access control.

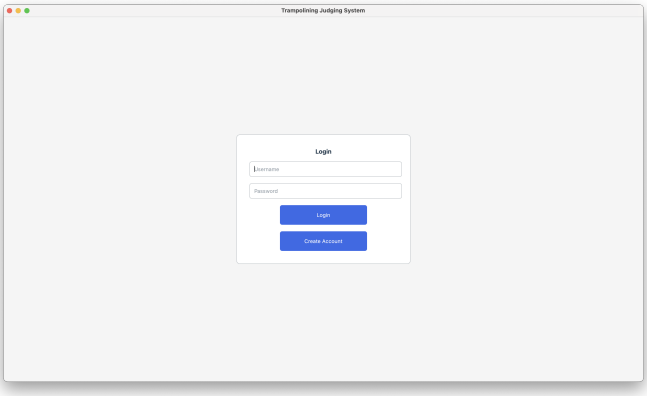

Figure 7: Login Page Interface

RegisterPage Enables new users to register by creating an account, capturing user details, and storing them securely.

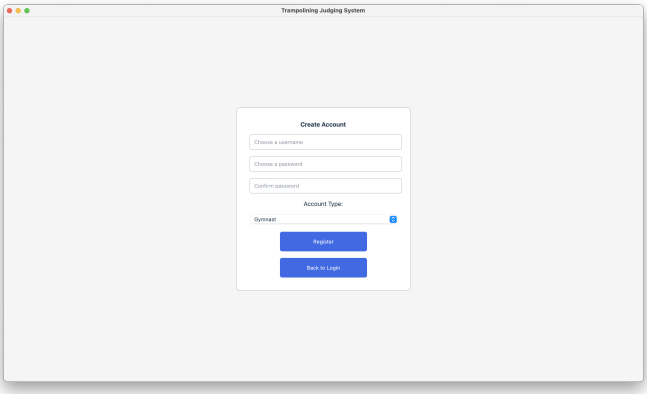

Figure 8: Register Page Interface

HomePage Serves as the main landing page after login, providing navigation to key functionalities including video uploads, data analysis, and profile management.

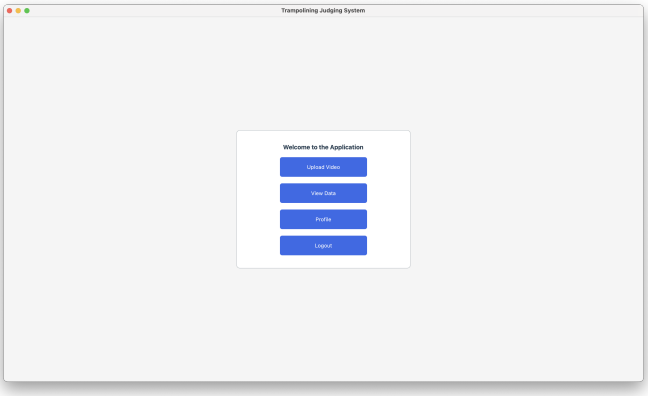

Figure 9: Home Page Interface

DataPage Displays user-specific data and analytics, enabling the viewing and exporting of performance metrics and historical data.

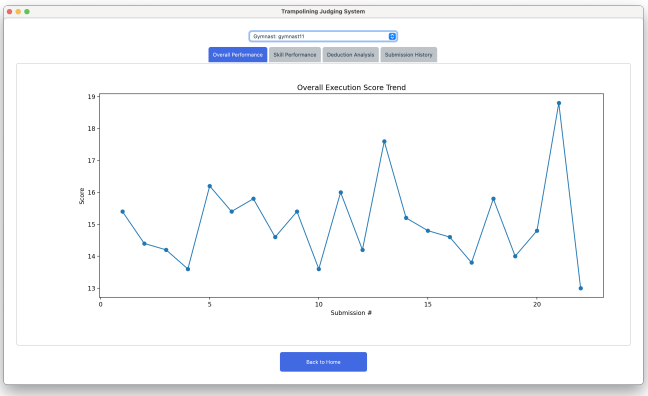

Figure 10: Data Page - Scores Over Time

Figure 11: Data Page - Average Deductions per Skill

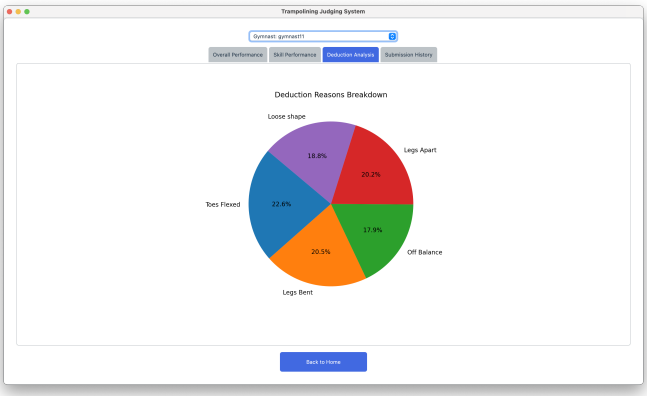

Figure 12: Data Page - Most Common Faults

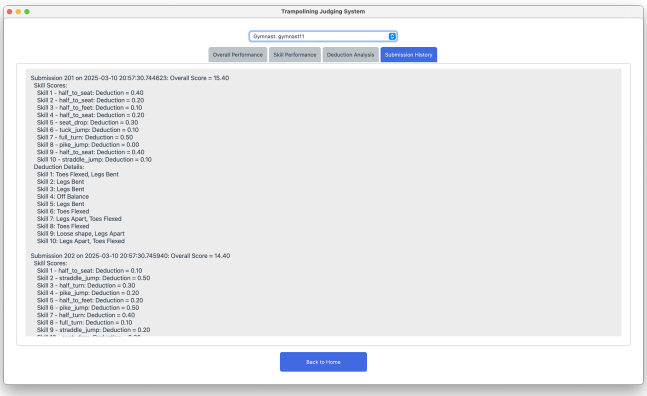

Figure 13: Data Page - Submission History

ProfilePage Allows users to view and manage their personal profile details, update credentials, and view connected coaches or gymnasts.

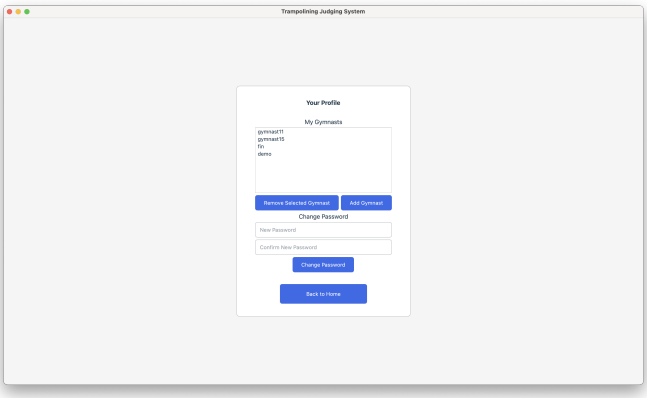

Figure 14: Profile Page (Coach)

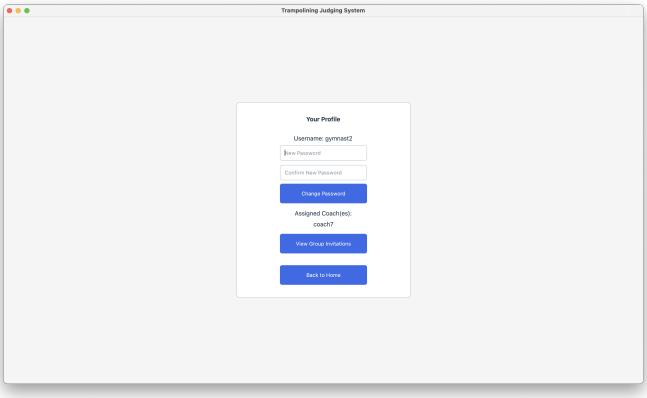

Figure 15: Profile Page (Gymnast)

GymnastSearchPage Provides functionality for coaches to search and manage gymnast connections, facilitating better team management.

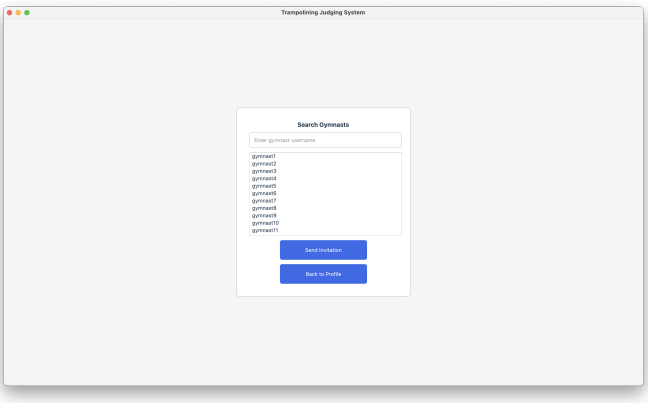

Figure 16: Gymnast Search Interface

InvitationsPage Displays pending invitations for gymnasts to join coaching groups, allowing acceptance or rejection of these invitations.

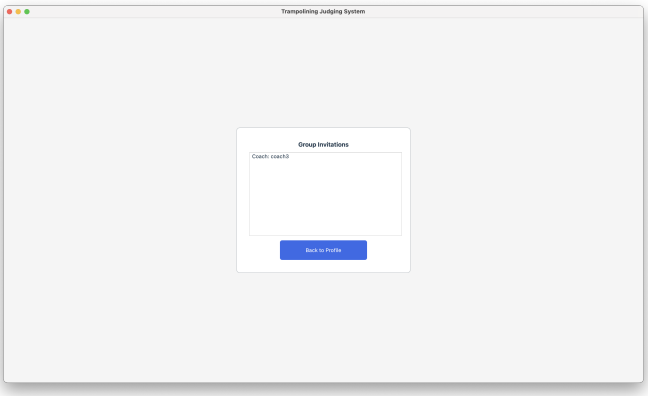

Figure 17: Gymnast Search Interface

UploadPage Enables users to upload trampoline routine videos directly from their desktop for subsequent analysis.

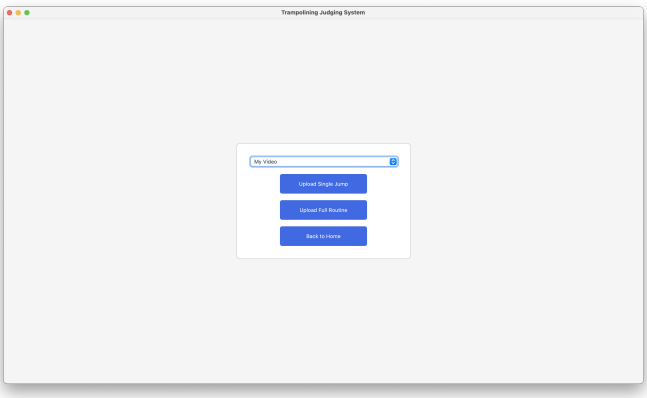

Figure 18: Upload Page Interface

DisplayScorePage Shows detailed analysis and scoring of uploaded trampoline routines, providing visualizations and score breakdowns.

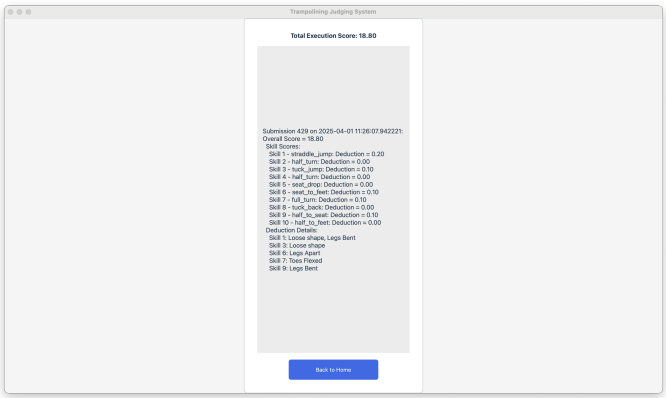

Figure 19: Display Score Interface

11.2.3 Integration with Backend

PyQt5 facilitated smooth backend integration using the signals and slots mechanism, allowing asynchronous operations without freezing the UI, critical for video analysis and processing tasks.

11.3 Integration and Workflow

Bringing everything together, the typical workflow for using the system is:

1. User logs in (coach or gymnast).
2. User goes to Upload, selects a video of a routine.

3. System processes the video: runs pose estimation (frame by frame), skill segmentation, classification for each skill, and execution scoring for each skill.

4. System stores the results and displays them to the user once ready.

5. User reviews the results. If a coach, they might share the results with the gymnast or plan training based on it. If a gymnast, they directly take the feedback for self-improvement.

The entire process from upload to result display for a single routine (roughly 10 skills, $\tilde{1}0$ seconds of video at 30 fps) was around 8 seconds in the prototype environment. Pose estimation was the most time-consuming part. This processing time is acceptable in a training context (the gymnast can wait a minute after practice to get the feedback). It's not real-time yet, but with optimisation or more computing power (using a GPU server), it could approach real-time.

The integration was tested with multiple videos in succession, ensuring the database correctly handled multiple entries and the UI could display each. I also tested error cases: e.g., uploading a video with an unsupported format. Logging was put in place on the server side to record each step, which helped in debugging issues.

In conclusion, the database and application development tied together the project into a usable system. The design considerations for multi-user support and the interactive features for reviewing results were important to make the technology accessible and valuable in practice. With the system implemented, the next step was to evaluate its performance and effectiveness, which is discussed in the following chapter.

12 Evaluation, Results & Testing

This section presents a detailed evaluation of the developed AI trampoline judging system, focusing on quantitative performance metrics and qualitative user feedback analysis. Comprehensive data is presented alongside illustrative visualisations and detailed tables to facilitate a clear understanding of the system's effectiveness and reception by its intended users.

12.1 Quantitative Evaluation

Building on the training, validation, and hyper-parameter tuning process described earlier, this section details the evaluation of the model using both cross-validation and a held-out test set.

12.1.1 Cross-validation Performance Analysis:

Within the training set, a 5-fold cross-validation scheme was applied exclusively for hyper-parameter tuning. For each hyper-parameter configuration, performance metrics were aggregated by averaging across folds. Although the individual folds exhibited variability due to random train/validation splits (with accuracies ranging from approximately 77% to 94%), the average validation loss of **0.4495** and accuracy of **85.70%** provided a robust estimate for selecting the optimal configuration. This averaging approach minimises bias from any single split and helps ensure that the chosen hyper-parameters generalise well.

12.1.2 Test Set Evaluation:

Performance evaluation on the entirely unseen testing dataset yielded an accuracy of **87.18%**, surpassing the cross-validation baseline. This confirms strong generalisation to novel instances, crucial for operational deployment. The improvement in accuracy compared to cross-validation indicates robust learning, suggesting that the selected hyper parameters were well-tuned. This high accuracy allows for a high level of trust that the correct skill was identified when passing results to the next stage of the overall scoring model.

12.1.3 Detailed Performance Metrics and Error Analysis

The confusion matrix in Table 2 explicitly reveals that most misclassification errors occurred between closely related skill classes, notably affecting pike jumps with straddle jumps. This suggests that the current features or model architecture may lack sufficient granularity to differentiate highly similar skills, such as subtle distinctions in limb positioning or trajectory. Conversely, perfect or near-perfect classification of several classes indicates clear and distinct learned representations for these skills. The high precision and recall metrics (weighted averages of 0.88 and 0.87, respectively) support overall robust classification performance, though the errors in specific classes suggest potential improvements via additional training data or refined feature extraction methods.

	full_turn	half_to_feet	half_to_seat	half_turn	pike_jump	seat_drop	seat_to_feet	straddle_jump	tuck_back	tuck_jump
full_turn	24	1	0	3	0	0	0	0	0	0
half_to_feet	0	12	0	0	0	0	0	0	0	0
half_to_seat	0	1	5	0	0	0	0	0	0	0
half_turn	1	2	0	24	0	0	0	0	1	0
pike_jump	0	1	0	0	14	0	2	6	0	0
seat_drop	0	0	0	0	0	29	0	0	0	0
seat_to_feet	0	0	0	0	0	0	31	1	0	0
straddle_jump	0	0	0	0	0	0	3	27	0	5
tuck_back	0	0	0	0	0	0	0	0	11	0
tuck_jump	0	1	0	0	0	0	1	1	0	27

Table 2: Confusion Matrix

Class	**Precision**	**Recall**	**F1-score**	**Support**
full_turn	0.96	0.86	0.91	28
half_to_feet	0.67	1.00	0.80	12
half_to_seat	1.00	0.83	0.91	6
half_turn	0.89	0.86	0.87	28
pike_jump	1.00	0.61	0.76	23
seat_drop	1.00	1.00	1.00	29
seat_to_feet	0.84	0.97	0.90	32
straddle_jump	0.77	0.77	0.77	35
tuck_back	0.92	1.00	0.96	11
tuck_jump	0.84	0.90	0.87	30
accuracy		0.87		234
macro avg	0.89	0.88	0.87	234
weighted avg	0.88	0.87	0.87	234

Table 3: Classification Report

The classification report in Table 3 corroborates these insights and highlights the variability in class-specific performance. The lower F1 score for pike jumps (0.76) emphasises particular challenges in distinguishing between visually and mechanically similar routines. This insight directs future model enhancement efforts towards feature engineering or data augmentation strategies that specifically target problematic skill classes.

12.1.4 Processing Time Evaluation:

In addition to classification accuracy, I evaluated the runtime performance of the system, as this has direct implications for usability in practice. The system

is implemented with efficiency in mind – using MediaPipe for fast pose estimation and a lightweight neural network for skill classification – and runs in near real-time. On a standard laptop CPU, the complete pipeline processes a typical 10-skill routine (about 20 seconds of video at 30 FPS) in approximately 8 seconds. This speed is well within practical limits: it means an athlete or coach gets feedback almost immediately after the routine is finished. In a competition setting, an automated score could be available moments after the athlete dismounts the trampoline, allowing it to complement human judges without causing any delays. The efficient performance also indicates that the model could potentially be deployed on edge devices or integrated into a tablet for gym use. In summary, the quantitative evaluation shows that the AI judging system is both accurate and fast, meeting the key requirements for a viable judging assistant.

12.1.5 Overall Judging Performance:

The overall model was evaluated using a small external dataset consisting of full 10-skill routines. Each video in the dataset was independently scored by two qualified judges to establish a reliable benchmark. To determine accuracy, each video was processed through the system, and the system's score was considered accurate if it fell within one mark of the judges' combined scores. This margin was chosen based on the standard judging protocol, where individual judges scoring execution out of 10 must agree within 0.5 marks to produce a valid combined score; thus, doubling this for a total execution score out of 20 yields an acceptable variance of 1 mark.

The model achieved an accuracy of **82%** across a dataset of 50 full routines.

The primary source of error was the misidentification of specific skills. Further analysis confirmed this, as evaluations of execution accuracy for correctly identified skills showed the system achieved the correct result **90%** of the time, allowing a small margin of error of 0.1 marks per skill. Execution scoring relied on a rule-based approach, and errors typically arose due to noise in the MediaPipe skeleton extraction process causing incorrect joint readings, thus incorrectly triggering scoring thresholds.

12.2 Qualitative Evaluation

User feedback from gymnasts and trampoline coaches was gathered through informal interviews, focusing on system usability, trustworthiness, interpretability, and perceived value.

12.2.1 Expert Panel and Coach Feedback

Experienced trampoline coaches provided insightful feedback emphasising the accuracy and objectivity of the system:

> "The system was surprisingly accurate in spotting form errors. Initially, I doubted if the system could match human judgment, but the consistency was impressive." (Coach A)

> "It's quite strict on toe pointing deductions—sometimes stricter than a human judge might be—but overall, it's a very useful tool." (Coach B)

Feedback indicated a high degree of initial skepticism, gradually replaced by confidence in the system after direct interaction and validation against human judgments. Suggestions included refining thresholds to better align with nuanced human judgment standards.

12.2.2 Gymnast User Feedback

Gymnasts also responded positively, highlighting the system's value in reinforcing training feedback:

> "I really liked how it pointed out the reasons for the score it was giving. This helped me trust the score it gave me more" (Gymnast 1)

> "One deduction about my legs being slightly apart seemed too strict, but everything else matched my performance accurately." (Gymnast 2)

These statements underscore the interpretability and educational value of the system's automated feedback.

12.2.3 Usability and Interface Interpretability

Users highly rated the desktop application's ease of use and clarity. Uploading routines and accessing detailed scoring and visual feedback was intuitive and straightforward.

12.2.4 Trust and Value Perception

A critical qualitative metric was trust in the system's judgments. Users reported an increase in trust over time as they verified the accuracy of the system's feedback. Coaches indicated willingness to integrate the system into regular training sessions, particularly for preliminary assessments and feedback reinforcement, although they recommended human oversight for critical decision-making contexts such as competitions.

12.3 Requirements Testing

To ensure that the system robustly met the predefined requirements outlined in Section 5, targeted testing was systematically performed during each sprint. Each requirement was individually verified, resulting in a comprehensive assessment that reinforced confidence in system compliance.

12.3.1 Functional Requirements Verification

Data Collection (FR1): Testing confirmed that the video dataset collected from Warwick Trampolining adequately represented diverse trampoline skills and execution variations. Routine analyses consistently reflected the diverse scenarios specified in FR1.1 and FR1.2.

User Interface (FR2): The GUI was rigorously tested through user interactions, meeting requirements FR2.1, FR2.2, and FR2.3. Uploading routines and accessing scores and detailed feedback operated as intended without usability issues.

Pose Estimation (FR3): Validation tests using MediaPipe confirmed successful integration of 3D pose estimation, with generated data reliably feeding into machine learning processes, thus satisfying FR3.1 and FR3.2.

Skill Recognition (FR4): Quantitative testing demonstrated an overall skill recognition accuracy of 87.18%, exceeding the FR4.2 minimum threshold of 80%.

Temporal Action Localisation (FR5): Temporal accuracy testing verified precise skill timing identification, essential for accurate scoring and fully compliant with requirements FR5.1 and FR5.2.

Execution Scoring (FR6): The system's execution scoring accuracy tests demonstrated scores consistently within ± 0.5 points of qualified human judges for 90% of correctly identified skills, successfully meeting requirement FR6.2.

12.3.2 Non-Functional Requirements Verification

Performance and Efficiency (NFR1): Performance benchmarks demonstrated average processing completion for standard 10-skill routines in approximately 8 seconds, comfortably within the one-minute limit set by NFR1.1.

Usability (NFR2): User feedback through surveys and interviews indicated high intuitiveness and interpretability of the interface, aligning strongly with NFR2.1 and NFR2.2.

Accuracy and Reliability (NFR3): Reliability testing confirmed overall scoring consistency closely matched human judges (within ± 0.5 points), meeting the stringent accuracy criterion set by NFR3.1. Additionally, robustness tests demonstrated reliable skill recognition across variations in camera angles and lighting conditions, fully satisfying NFR3.2.

Maintainability and Extensibility (NFR4): Code reviews and architecture assessments confirmed the system's maintainability and ease of extending databases and scoring rules, achieving compliance with NFR4.1.

Scalability (NFR5): Scalability testing through simulated user load showed that the system could accommodate increased routine processing without performance degradation, meeting NFR5.1.

This structured verification of requirements demonstrates a high level of compliance and provides assurance of the developed system's comprehensive functionality and reliability as originally intended.

12.4 Summary of Evaluation Outcomes

The quantitative evaluation rigorously assessed the AI trampoline judging model through multiple validation methods, achieving robust and reliable performance. Cross-validation produced an average accuracy of 85.70%, demonstrating stable model behavior across different data splits and validating the hyperparameter selection. When tested against a completely unseen dataset, the model's accuracy improved further to 87.18%, signifying strong generalization capabilities and effectiveness in accurately classifying skills.

Detailed error analysis via confusion matrices and classification reports highlighted the model's precision in distinguishing most skill classes, yet revealed specific challenges in differentiating visually similar movements, particularly between pike and straddle jumps. The lower recall (61%) and corresponding F1-score (0.76) for pike jumps indicate potential areas for improvement, suggesting targeted data augmentation or refined feature extraction could enhance accuracy.

Moreover, the system exhibited near real-time performance, processing a full 10-skill routine in approximately 8 seconds on standard hardware. This efficiency meets practical competition requirements, allowing rapid feedback and integration into live judging scenarios.

In an external evaluation of full routines, judged by professional standards, the overall system demonstrated a commendable accuracy of 82%, mainly impacted by occasional skill misclassification and minor execution scoring discrepancies arising from pose estimation inaccuracies. However, when skills were correctly classified, execution scores were accurate 90% of the time, underscoring the reliability of the scoring methodology.

Qualitative evaluations revealed that expert trampoline coaches and gymnasts initially expressed scepticism but quickly developed trust in the system's consistency and accuracy. Coaches appreciated the system's objective judgments and recommended minor refinements to better match nuanced human judgment. Gymnasts found the detailed scoring feedback valuable for training, noting enhanced interpretability and trustworthiness of the automated assessments. The user interface received highly positive feedback for usability and clarity, indicating strong practical applicability.

Comprehensive testing of both functional and non-functional requirements demonstrated high compliance across all predefined criteria. Functional testing verified successful data collection, accurate pose estimation, robust skill recognition (87.18% accuracy), precise temporal localisation, and reliable execution scoring. Non-functional testing confirmed efficient performance (processing within 8 seconds), high usability, strong reliability (± 0.5 points accuracy compared to human judges), maintainability, and scalability. These outcomes collectively establish the system's readiness and effectiveness for real-world competitive and training environments.

13 Discussion and Limitations

The development and evaluation of the judging system provides several points of discussion in terms of technical achievements, limitations, and implications. In this chapter, I interpret the results from Chapter 12, compare the approach to related work, discuss the significance of the system for the sport of trampoline gymnastics, and examine ethical and practical considerations of deploying such technology.

13.1 Interpretation of Findings

The project demonstrated that combining computer vision and expert knowledge rules is a viable approach to partially automate trampoline scoring. The skill classifier's success indicates that even with relatively modest data, a machine learning model can learn to distinguish complex aerial manoeuvrers. This underscores the power of pose-based features; by reducing raw video to keypoint motion, I captured the essence of each skill in a way that the model could parse, which aligns with findings in similar research. My $\tilde{8}7\%$ accuracy is respectable, although there is room for improvement. Perhaps with more training data, accuracy could increase. One lesson learned is that data augmentation and synthetic data could be key – e.g., generating slightly altered motion sequences to increase variety, or using physics-based models to simulate some trampoline motions might expand the dataset cheaply.

The rule-based execution scoring, while not perfect, provided consistent application of judging criteria. The alignment to within about 0.5 points of human judges suggests that explicit modelling of deductions can be effective. This result is encouraging because it means that even without a huge dataset to train an execution scoring model, I can hand-craft a system that is useful. However, the discrepancies observed (like over-penalising toe flexing) highlight a limitation: the rule system is only as good as the knowledge I encode and the sensor information I have. For example, lacking direct measurement of horizontal displacement limited my ability to mimic that aspect of judging. One could incorporate additional sensors (like a second camera for top-down view or an actual trampoline bed sensor, which in competitions measures time-of-flight and displacement). Integrating those would greatly enhance the completeness of the scoring.

From a user perspective, the high satisfaction suggests that, despite any errors, the system delivered value. This indicates that even if an AI judge is not perfect, as long as it provides mostly correct feedback and is transparent, coaches and athletes see benefit in it. The transparency of my system (explaining deductions) was likely crucial for this acceptance. If I had a black-box model say "Execution score 16.0" with no explanation, users might be more wary. By listing each deduction and linking to a known rule, I gave users a narrative that they could accept or contest. This approach aligns well with the concept of explainable AI, which is particularly important in domains like sports or medicine where users demand to know the reason behind an AI's decision.

13.2 Comparison to Related Work

Comparing my approach to Fujitsu's Judging Support System (JSS) for gymnastics: JSS heavily relies on advanced hardware (multiple 3D Lidar sensors, etc.) to get precise 3D models of the gymnast. In contrast, my approach was done with a single camera and purely software-based pose estimation. Obviously, JSS likely achieves higher accuracy and can do things like real-time multi-angle replay, but it's a multi-million dollar system. My system is low-cost and more accessible, at the expense of some precision. Interestingly, JSS in artistic gymnastics focuses on identifying elements and providing angles for judges to use in execution scoring, rather than fully automating execution. My project attempted to go a step further in automation of execution (albeit for trampoline, which has a bit simpler execution criteria in some ways). In literature, the sensor-based trampoline difficulty judging by Woltmann et al. (2023) achieved very high skill recognition (96%). That confirms that given the right sensing (IMUs on the gymnast's body), classifying skills is almost solved. However, their approach doesn't address execution. One could imagine combining an IMU-based difficulty judging system with a vision-based execution judging like mine for a complete package.

13.3 Limitations

While my system performed well in tests, it has clear limitations:

- Generality: I focused on a subset of trampoline skills (mostly individual flips, twists, jumps). I did not fully handle the large number of skills available in high level trampoline. The system would need an extension and more training data to cover the full spectrum used in elite competitions.
- Environmental constraints: My pose estimation works with a clear side view of the trampoline. Different camera angles or poor lighting could degrade accuracy. I assumed a certain setup. In real competitions, multiple angles are used. My approach would need adaptation or multi-camera integration to be robust in different settings.
- Lack of training data: The system was bottlenecked by the amount of data that could be collected from Warwick University trampolining sessions. This caused the model to need to be limited to a reduced number of skills, and did not allow for a hybrid data-driven and rule-based model to be created for the execution scoring.
- Rule Tuning: I tuned thresholds on a limited set of data. Different body types (short vs tall athletes) might need some adjustment in how angles map to visual faults, though my normalisation attempts to mitigate that. Ideally, those thresholds would be validated on a large set of routines scored by judges to ensure consistency with judging norms.
- Under-estimation of data collection and labelling: One of the reasons for the lack of training was the under-estimation of how long data collection and labelling. This matched with the time constraint of the project set back the project scope.

13.4 Project Reflections

From a project management perspective, my agile approach allowed me to pivot when needed, like when I realised that an ML execution scorer was infeasible with my data, I doubled down on rules. The iterative testing with real users (coaches, etc.) was invaluable; it's clear that involving end-users early leads to a better product.

14 Future Work

The success and limitations of the system open up multiple avenues for future development and research. This chapter outlines a number of improvements and extensions that could enhance the system's accuracy, scope, and usability. These suggestions build upon the current work and take into account both the feedback received and the evolving landscape of AI in sports officiating.

14.1 Enhanced Classification Model and Data Expansion

One clear path is to improve the skill classification model. This can be done by:

- Gathering more data: Increase the dataset of labeled trampoline skills. Collaborating with gymnastics clubs or using public competition footage could yield more examples. A larger and more diverse dataset would likely boost the model's accuracy and robustness.
- Augmentation Techniques: Beyond horizontal flipping, you could simulate slight camera angle variations, noise, or different speeds to make the model invariant to those factors. Use of Generative Adversarial Networks (GANs) to synthetically generate plausible trampoline motion data might also be explored, although ensuring physical realism is a challenge.

14.2 Hybrid Execution Model

The expansion of the collected dataset presents an opportunity to explore the implementation of a hybrid execution scoring model, integrating both data-driven and rule-based methods. A hybrid approach could substantially enhance the system's sensitivity, enabling it to identify and evaluate more nuanced deductions—such as slight form errors, minor body movements, or subtle alignment issues—which may currently be overlooked by a purely rule-based approach.

In particular, a data-driven model trained on extensive annotated video examples could automatically learn to recognise subtle execution errors through pattern recognition and anomaly detection techniques. This capability would complement the rule-based system, which excels at capturing clearly defined and easily quantifiable faults but may miss deviations too subtle or context-specific for explicit rule codification.

However, implementing a hybrid approach does introduce potential trade-offs, notably concerning the transparency and interpretability of the scoring system. One strength of the current rule-based model is its explicit transparency, users can clearly understand the rationale behind every deduction. In contrast, machine learning models, particularly deep learning approaches, may act as a "black box," obscuring the specific reasons behind scoring decisions. This opacity could negatively impact user trust, as athletes and coaches might question the fairness or consistency of judgments if explanations for deductions become less explicit.

To address these concerns, a hybrid model implementation should prioritise interpretability. Techniques such as explainable AI (XAI) methods could be integrated, providing clarity about why specific deductions were applied. Clear explanations accompanying data-driven deductions would be essential to maintaining user confidence and trust.

Overall, while a hybrid execution scoring model promises significant improvements in accuracy and sensitivity to subtle performance nuances, careful consideration must be given to balancing enhanced performance with transparency and interpretability, ensuring sustained user acceptance and confidence in the system.

14.3 Integrating Additional Sensors and Metrics

To enhance the robustness and comprehensiveness of the system, integrating additional sensors and metrics could significantly improve the assessment of performance factors such as horizontal displacement, time of flight, and reduce overall noise in the detection process. However, these enhancements may come with trade-off's, notably reducing the accessibility and ease of use of the current setup. A potential solution would be to develop an alternative, specialised version of the system designed specifically for professional competition environments where accuracy and comprehensiveness are paramount.

Additional sensor and metric integration could include:

- Multiple Camera Angles: Introducing multiple camera perspectives, such as adding a top-down view, can directly address the accurate measurement of horizontal displacement (travel) on the trampoline bed. Utilizing stereo vision, where two cameras capture simultaneous video from different angles, allows for precise three-dimensional reconstruction of the gymnast's movements without the need for specialised infrared equipment. This method significantly enhances the accuracy of spatial measurements, allowing accurate triangulation of 3D coordinates for key points and thereby calculating actual distances and angles more reliably.
- Trampoline Bed Sensors: Integrating sensors directly onto the trampoline bed can provide objective data on performance metrics like Time of Flight (ToF). Such sensors, already common in professional competitions, measure precise contact and flight times, which are critical scoring components under Fédération Internationale de Gymnastique (FIG) rules. Incorporating trampoline bed sensors would not only provide an official measure of bounce height and timing but also enhance routine segmentation accuracy by precisely identifying points of contact and flight phases.

14.4 Improved User Interface and Analytics

From a software perspective:

- More Analytics: Provide users with deeper insights, such as progress charts in the build up to a specific competition. Possibly incorporate video management where users can tag specific attempts and track improvements.

- Cloud Platform: Turn the system into a web service where users worldwide can upload their routine videos and get scores. This would require good scaling. This future direction brings its own engineering challenges (concurrency, user management at scale, etc.), but from a research perspective, it would generate a lot of data that could further improve the models.

14.5 Collaboration with Governing Bodies

Future development should consider working with official bodies (like FIG or national gymnastics federations):

- They could provide data (videos from competitions with official scores) for training and validation, which would greatly improve the system.
- Ensuring alignment with the official Code of Points and any rule changes: a collaboration would mean the AI could be updated in sync with any rule updates or interpretations.
- Pilots in competitions: Perhaps testing the system as a shadow judge in junior competitions to see how it compares and refining it. Over time, this could build trust in the system.

15 Conclusion

This project set out to create a coaching aid system for trampolining routines, leveraging AI to provide consistent skill recognition and execution feedback and removing the subjectivity of trampoline judging. Throughout the development and evaluation, it was demonstrated that modern computer vision and machine learning techniques can indeed assess trampoline routines with a level of accuracy that approaches human judgment.

A pipeline was built that detects a gymnast's movements via pose estimation, identifies the skills performed using an LSTM-based classifier, and evaluates execution through a rule-based deduction model aligned with FIG standards. The integrated system was packaged into a user-friendly application, enabling practical use by coaches and gymnasts.

Key achievements include:

- Achieving over 87% accuracy in automatic skill recognition on unseen data – surpassing earlier attempts in literature
- Achieving a 82% accuracy in execution judging on unseen data.
- Implementing an execution scoring component that generally mirrors human judges closely (within 0.5 points on average)
- Delivering feedback in an interpretable format that was well-received by end users.

Findings from user testing confirmed that the system's consistency and objectivity are valuable, and that providing explainable feedback greatly increases user trust in the AI's decisions. Coaches found that it could reinforce their coaching by catching details and providing quantifiable analysis, while gymnasts enjoyed the immediate, detailed insight into their own performances.

Through this project, I also identified limitations – notably, the need for more training data to broaden skill coverage and the challenges of capturing all nuances of execution with simple rules. I addressed some limitations by focusing on transparent design and by laying out clear future improvements such as hybrid models and additional sensor integration.

The implications of this work are significant for the sport of trampoline gymnastics: it shows a pathway to more objective judging and enhanced training methodologies. By having an ever-watchful AI "coach" that never tires or biases, athletes could potentially improve faster and competitions could be judged more fairly. However, it also highlighted that technology must be introduced thoughtfully, with attention to the human elements of the sport.

In conclusion, the project achieved its primary goal of creating a functional AI coaching aid for trampoline routines and demonstrated its potential benefits. It bridges a gap between theoretical research and practical application in sports and provided an approach that successfully reduces subjectivity in

judging. With further refinement and collaboration, such a system could transform how trampolining and similar judged sports are trained and judged, complementing human expertise with reliable, data-driven assistance. This work contributes a stepping stone toward that future, indicating that the fusion of sports science and AI holds great promise in enhancing athletic performance and fairness in competition.

References

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001), 'Manifesto for agile software development'. Accessed: 9 April 2025.
URL: *https://agilemanifesto.org/*

Cohen, D., Lindvall, M. and Costa, P. (2004), 'An introduction to agile methods', *Advances in Computers* **62**, 1–66.
URL: *https://www.sciencedirect.com/science/article/pii/S0065245803620012*

Connolly, P. W., Silvestre, G. C. and Bleakley, C. J. (2017), 'Automated identification of trampoline skills using computer vision extracted pose estimation', *arXiv preprint arXiv:1709.03399* .
URL: *https://arxiv.org/abs/1709.03399*

Díaz-Pereira, M. P., Gómez-Conde, I., Escalona, M. and Olivieri, D. N. (2014), 'Automatic recognition and scoring of olympic rhythmic gymnastic movements', *Human Movement Science* **34**, 63–80.

Fujitsu (2023), 'Judging support system co-development with fig'. Accessed: 9 April 2025.
URL: *https://www.fujitsu.com/global/themes/data-driven/judging-support-system/*

Fédération Internationale de Gymnastique (2024), 'Code of Points: Trampoline Gymnastics 2025–2028'. Accessed: 9 April 2025.
URL: *https://www.gymnastics.sport/publicdir/rules/files/en*$_1$.1%20 − %20*TRA*%20*CoP*%202025 − 2028.*pdf*

Helten, T., Brock, H., Müller, M. and Seidel, H.-P. (2011), 'Classification of trampoline jumps using inertial sensors', *Sports Engineering* **14**(2-4), 155–164.

International Olympic Committee (2024), 'Trampoline gymnastics'. Accessed: 2024-04-09.
URL: *https://olympics.com/en/sports/trampoline-gymnastics/*

Okamoto, L. (2024), 'Transparent and objective ai scoring system for competitive diving', `http://arks.princeton.edu/ark:/88435/dsp01jh343w653`. Accessed: 9 April 2025.

Rubin, K. S. (2013), *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, Addison-Wesley.
URL: *https://www.pearson.com/us/higher-education/program/Rubin-Essential-Scrum-A-Practical-Guide-to-the-Most-Popular-Agile-Process/PGM332852.html*

Stellman, A. and Greene, J. (2014), *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*, O'Reilly Media.
URL: *https://www.oreilly.com/library/view/learning-agile/9781449331924/*

Woltmann, A., Henrich, D. and Göb, S. (2023), 'Sensor-based jump detection and classification with machine learning in trampoline gymnastics', *German Journal of Exercise and Sport Research* **53**, 435–444. Accessed: 9 April 2025. **URL:** *https://link.springer.com/article/10.1007/s12662-022-00866-3*