From Octagon to Algorithm: A Knockout Approach to Building the Ultimate Fighting Classifier for Combat Sports Action Recognition

Gary Gau Ku
Master of Engineering - Data Science
University of California, Riverside
gaugarygrayston@gmail.com
SID: 862397381
August, 2024

Abstract: *The classification of combat sports movements from video data presents a complex challenge due to the sport's dynamic nature. This research aims to address these challenges by integrating advanced pose estimation techniques with machine learning models to accurately classify Mixed Martial Arts (MMA) techniques. The research involves recording MMA bouts using multiple cameras in a controlled environment, manually labeling the video frames with specific combat actions, and extracting pose landmarks. The study leverages the SSD MobileNet model for people detection and MediaPipe Pose Estimator for extracting key anatomical landmarks. Feature engineering includes calculating distances and angles between key points to enhance the data representation. These features are used to train two machine learning models: a Feedforward Neural Network (FNN) and a Long Short-Term Memory (LSTM) network. The FNN treats each frame independently, while the LSTM captures temporal dependencies in sequences of frames. The LSTM model significantly outperformed the FNN, achieving an accuracy of 93% compared to the FNN's 41%. These results validate the hypothesis that incorporating temporal data enhances model accuracy in video-based classification tasks. However, limitations such as manual labeling biases and the need for more controlled data collection are acknowledged. Future research should explore advanced machine learning models and real-time applications, aiming to generalize the methodology to various sports and environments. This research contributes to the growing field of sports analytics by developing a robust framework for video-based action classification, with practical implications for athletes, coaches, and broadcasters.*

Introduction

Overview

The classification of combat sports movements from video data presents a multifaceted challenge due to the intricate and rapid interactions between fighters, encompassing diverse techniques, unpredictable movements, and dynamic exchanges. These interactions, characterized by their high speed, variability, and complexity, pose significant obstacles to traditional image processing and machine learning techniques, which often struggle to accurately capture and analyze the nuances of such dynamic environments. This study seeks to address these challenges by leveraging advanced pose estimation methodologies combined with robust feature extraction techniques and sophisticated machine learning models to accurately classify various combat sports actions.

In recent years, advancements in the field of computer vision and artificial intelligence (AI) has opened up new avenues for analysis, performance enhancement, and entertainment. Combat sports in particular, with their complex and fast-paced nature, stand to benefit immensely from AI-driven analysis. As a combat sports competitor myself, I recognize the value of an AI system capable of recognizing and analyzing actions in real-time, providing feedback that can assist both athletes and trainers. Such a system could democratize access to high-level coaching, making the sport more accessible to aspiring fighters who may not have the resources to afford a human coach. Moreover, this AI could serve as a powerful tool for coaches, complementing their expertise with precise, data-driven insights.

Furthermore, broadcasters and content creators could leverage this technology to deliver more engaging and informative content to audiences, potentially transforming the way MMA is consumed by fans. The transition from image classification to video classification introduces new challenges, including the need to account for temporal dependencies between frames, variability in camera angles, lighting conditions, and the complexity of fighter movements. Unlike static images, where the classifier only needs to focus on a single frame, video classification requires the model to understand the context and continuity of movements over time.

This project draws upon foundational work in the field of sports analytics, including my own preliminary investigations, which have explored the potential of machine learning models in sports-related classification tasks. By focusing specifically on Mixed Martial Arts (MMA)—a sport chosen for its prevalence and complex dynamics—this study aims to extend these initial findings. I plan to utilize and incorporate models and tools from many studies in the field of computer vision and machine learning in order to build a dynamic video-based classification framework that mirrors the real-time decision-making processes in MMA competitions. This

approach allows us to address the unique challenges posed by the sport's fast-paced and unpredictable nature.

Through this research, I aim to make a meaningful contribution to the field by developing methods that accurately recognize and classify actions in combat sports. By doing so, this work not only advances the technical capabilities of AI in sports but also opens up new possibilities for athletes, coaches, and the sport itself. By integrating sophisticated models capable of capturing both spatial and temporal features, the proposed research ensures accurate and reliable classification across different scenarios, potentially leading to a revolution in how combat sports are trained for, performed, and viewed.

Literature Review

The integration of image classification into sports analytics represents a burgeoning field, particularly salient in high-impact sports such as Mixed Martial Arts (MMA). Understanding the complex body movements within such sports offers profound insights into performance strategies and training regimes. Historical applications of machine learning to classify sports imagery have set a foundational framework for this inquiry.

For instance, Minhas et al. (2019) utilized a neural network to classify sports video frames into long shot, medium shot, close-up shot, and crowd shot categories with an impressive 94% accuracy. Their study employed transfer learning to adapt the general AlexNet model into a specialized model for sports image classification, highlighting the effectiveness of transfer learning in developing accurate classifiers for specific sports-related tasks.

Similarly, Podgorelec, Pecnik, and Vrbancic (2020) conducted a study where they used a convolutional neural network (CNN) to classify sports images into four different ball sports (football, soccer, rugby, and field hockey). By employing transfer learning to adapt the VGG19 model into a specialized sports image classifier and further optimizing it through hyperparameter tuning, they achieved an accuracy of 81% with an F1 score of 82%. These studies underscore the potential of CNNs and transfer learning in sports image classification, providing a strong foundation for extending these techniques to video-based classification tasks in MMA.

In the realm of combat sports, Kasiri-Bidhendi et al. (2015) developed a framework that achieved a remarkable 96% accuracy in classifying overhead images of boxer punches. This study is particularly relevant to the current project, as it highlights the feasibility of applying machine learning techniques to combat sports, where the dynamic nature of the sport presents unique challenges in terms of accurate classification. However, this study was limited to classification of punching actions and restricted to only overhead images.

My own previous research focused on applying these techniques to MMA, where I webscraped and labeled over 4,000 images of MMA bouts and utilized transfer learning to create custom CNNs based on the AlexNet and VGG19 models. The AlexNet model, with a validation accuracy of 75%, showed promise, and the results aligned with previous research on sports image classification.

However, despite promising results, I hypothesized that the static nature of image classification proved insufficient for the dynamic sequences inherent in MMA, prompting a shift towards video-based data and models that can effectively integrate temporal dimensions for enhanced accuracy. I believed that static image classification approach proved to be suboptimal for classifying MMA techniques due to the dynamic movement patterns inherent in the sport. To address this limitation, my current project seeks to leverage video-based data and employ a model that can leverage the time component of video data to create a more accurate classifier. Unfortunately, because the Alexnet and VGG19 model were trained for static image classification, I could not reuse the transfer learning methods I previously employed, and thus it was necessary to develop a custom method for feature extraction.

The feature extraction methodology developed for the current study builds on several key advancements in the fields of pose estimation and action recognition. Previous studies have demonstrated the potential of pose estimation for analyzing human activities in sports, with a growing body of research focusing on improving the accuracy and applicability of these techniques in real-world scenarios (Shotton et al., 2011). The MediaPipe Pose Estimator, developed by Lugaresi et al. (2019), has emerged as a leading tool for anatomical landmark detection. MediaPipe's robust performance in tracking human body landmarks across frames has been validated in various settings, including sports analytics, where it has proven effective in capturing detailed body movements (Lugaresi et al., 2019). Its application in this study is critical for extracting precise pose landmarks, which serve as the foundation for subsequent feature engineering and classification tasks. However, the was the limitation of the MediaPipe Pose Estimator was that it was designed to work on a single person. As MMA is a two-person sport, a method to adapt the pose estimator for use with multiple people was a necessity.

To address this limitation, a people detection model was incorporated to create bounding boxes around each fighter in the video. This allowed for the use of a method for the pose estimator to work on one fighter at a time until pose estimations for both fighters were generated. The SSD MobileNet model, introduced by Howard et al. (2017), has been widely recognized for its efficiency and accuracy in real-time object detection, making it an ideal choice for detecting and isolating individuals in video frames. This model's ability to balance speed and accuracy is particularly relevant in scenarios involving fast-paced movements, such as those seen in MMA bouts, where the rapid and often unpredictable actions of fighters necessitate a model that can operate in real-time without sacrificing precision.

In addition to these tools, the literature on human motion analysis emphasizes the importance of capturing both spatial and temporal features to fully understand complex movements. Ye et al. (2013) review several methodologies for human motion analysis, highlighting the significance of spatial features, such as distances and angles between body parts, in capturing the essence of movement dynamics. This study builds on these insights by engineering relative position and angular feature into the dataset, thereby enhancing the ability to differentiate between various MMA techniques.

To predict and classify MMA actions effectively, the choice to use an LSTM network was inspired by its proven effectiveness in handling sequential data dependencies, essential for the fluid motions observed in sports. LSTMs have been shown to be strikingly effective in capturing temporal dependencies in sequential data, making them well-suited for recognizing and classifying complex human actions in videos (Hochreiter & Schmidhuber, 1997; Ullah et al., 2018).

In fact, LSTMs and pose estimation have been shown to synergize particularly well for the purposes of human activity recognition. Noori et al. (2019) present a robust approach to human activity recognition by leveraging computer vision and pattern recognition techniques. Their method employs the OpenPose library to extract anatomical key points from RGB images, focusing on capturing the motion of these key points across consecutive frames. This extracted motion data is then processed using a Recurrent Neural Network (RNN) with Long Short-term Memory (LSTM) cells to recognize activities. The study demonstrates the effectiveness of this method, achieving an overall accuracy of 92.4% on a publicly available activity dataset. This performance surpasses conventional approaches like support vector machines, decision trees, and random forests, which only reached a maximum accuracy of 78.5%.

Coincidentally, the framework I propose for my study is also similar to one proposed by Cob-Parro et al. (2023) for deep learning based human action recognition. In their study, they proposed a pipeline where a video would be run through the SSD Mobilenet model with a Kalman filter to generate preprocessing bounding boxes. This would then be followed by the use of an LSTM model to recognize the actions of the person in the video. Applying the framework to several established datasets (such as KTH, WVU, IXMAS, and WEIZMANN) yielded models that were all least 99% accurate in classifying a number of different actions. The only difference between my framework and theirs is my use of a human pose estimator after attaining the bounding boxes with the SSD Mobilenet model. Thus, the results of this study also lends support to the framework proposed for my project.

Objective

Although literature on sports image analysis is relatively common, literature on combat sports image analysis is sparse. Despite extensive searches, I was unable to find any research pertaining to image classification applied specifically to combat sports or any relevant datasets. Therefore, the objectives of this study are twofold: (1) to develop a methodology for extracting features from combat sports video clips despite the single person limitations of current pose estimators, and (2) to test the hypothesis that temporal data can be leveraged to considerably boost model accuracy compared to static image analysis. The ultimate goal of this project is to develop a video classification model capable of accurately identifying and categorizing MMA actions from video sequences and contribute to the growing body of literature on combat sports analytics and provide a practical tool for stakeholders in the MMA community, including fighters, coaches, analysts, and broadcasters.

This study contributes to the ongoing development of pose estimation and action recognition methods by addressing the specific challenges associated with multi-person, dynamic environments like MMA. By combining SSD MobileNet and MediaPipe Pose Estimator with iterative checks for overlap and re-detection, the methodology developed in this research ensures accurate landmark extraction for each fighter, even in scenarios where traditional single-person pose estimators would struggle. The inclusion of these advanced techniques not only enhances the accuracy of pose estimation but also improves the overall reliability of the classification model, making it a valuable tool for analyzing complex combat sports scenarios.

Furthermore, this study extends the application of pose estimation in sports analytics by focusing on the classification of specific MMA techniques. The approach taken here—utilizing bidirectional long short-term memory (LSTM) networks—explores the potential of combining spatial and temporal features to improve the accuracy of movement classification. This approach not only serves as a benchmark for evaluating the effectiveness of incorporating sequential information but also provides insights into the relative strengths and limitations of different machine learning models in the context of sports analytics.

By advancing my previous methodology from static image classification to video-based analysis, this project aims to overcome the limitations of previous work and pave the way for more effective and applicable machine learning models in the field of sports analytics. The transition to video-based classification marks a significant step forward in the development of automated analysis tools for MMA, with the potential to revolutionize how the sport is understood, trained for, and consumed by audiences.

Methodology

Data Collection

The dataset used for this study was gathered through a meticulous setup designed to capture striking and grappling movements within an authentic training environment. Four high-definition cameras were strategically placed around the UFC Gym Octagon, ensuring comprehensive coverage from multiple angles. Positioned approximately 6 feet off the ground, these cameras were arranged on every other side of the Octagon's eight sides, with a distance of about 15 feet from the center. This setup allowed the cameras to capture the fighters' movements with minimal obstruction, crucial for generating clear and consistent footage.

The Octagon itself, with a diameter of roughly 30 feet, provided a controlled environment that closely mimics professional MMA fighting conditions. The fighters participating in this study were recorded as they engaged in various combat sequences, including striking and grappling. The placement of the cameras ensured that all angles of the fighters' movements were captured, offering a robust dataset for subsequent analysis.

To optimize the dataset for further processing and reduce computational demands, the videos were downsampled by selecting every other frame. This choice was driven by the need to balance the resolution of temporal details with computational efficiency. The downsampled frames maintained enough temporal granularity to capture critical moments of the fighters' actions while significantly reducing the volume of data to be processed.

The resolution of the videos was then standardized to 720p, ensuring consistency across the dataset. Each frame was further cropped to a 720 by 720 resolution, focusing the analysis on the fighters and minimizing extraneous background information. These preprocessed frames were subsequently recombined into video clips, each grouped by specific combat labels.

Video Preparation and Labeling

The video footage collected from the Octagon was meticulously processed to prepare it for pose estimation and landmark extraction. The primary objective of this stage was to segment the footage into clips, each clearly labeled according to the type of action taking place. The labeling process was manually conducted, allowing for precise categorization based on the following five predefined action labels:

- Punch: This label was assigned to frames where a fighter initiated a punch. The label encompassed the entire duration of the punch, from the moment the hand began extending from the body to strike the opponent until the hand returned to a neutral position.

- Kick: Frames were labeled as 'kick' when a fighter initiated a kick, covering the entire motion from the leg extending to strike the opponent to the leg returning to a neutral stance.
- Takedown: This label was applied to frames where a fighter initiated a takedown, characterized by a level-change maneuver aimed at grounding the opponent. The label covered the entire process from the initiation of the takedown until its completion or failure, depending on whether the opponent was successfully grounded or both fighters returned to a neutral standing position.
- Ground: Frames were labeled as 'ground' when one or both fighters were on the ground, signifying a grappling or ground-fighting situation.
- Not_engaged: This label was used for frames where both fighters were in a neutral stance, with neither fighter actively engaging in striking or grappling.

The manual labeling of these frames was a critical step in ensuring the accuracy of the dataset. However, it also introduced potential biases, as the labeling was conducted solely by the author. Future studies may consider involving multiple labelers to improve the robustness of the dataset and mitigate subjective biases.

The labeled frames were then organized into clips, each corresponding to one of the five action categories. These clips formed the basis for the subsequent pose estimation and landmark extraction processes.

Pose Estimation and Landmark Extraction Process

The extraction of pose landmarks from the labeled video clips was a pivotal step in this study, enabling the detailed analysis and classification of various MMA movements. Given the complexity and fluidity of MMA bouts, accurate pose estimation was crucial for capturing the nuances of fighters' interactions. The following detailed steps outline the methodology used to extract pose landmarks from each frame of the video clips:

1) **Person Detection and Bounding Box Creation**

   The first frame of each video clip was analyzed using the SSD MobileNet model, a deep learning-based object detection algorithm known for its efficiency in balancing speed and accuracy (Howard et al., 2017). SSD MobileNet was employed to detect the presence of fighters within the frame, generating bounding boxes around the detected individuals. To ensure the reliability of these detections, a confidence threshold was applied, filtering out low-confidence detections and retaining only the most reliable bounding boxes.

   Given the possibility of overlapping detections, where the same fighter might be detected twice due to close proximity or movement, only the two largest non-overlapping bounding

boxes were selected. This selection process was critical in distinguishing between the two fighters within the frame, setting the stage for accurate pose estimation.

2) Landmark Extraction for the First Fighter

With the bounding boxes established, the first box was used to mask the image, effectively isolating one fighter from the rest of the frame. This masking allowed for focused analysis, minimizing the influence of background elements or the other fighter.

The MediaPipe Pose Estimator, an advanced tool for real-time pose estimation (Lugaresi et al., 2019), was then applied to estimate the anatomical landmarks within the masked region. MediaPipe was chosen for its proven effectiveness in tracking human poses, even in complex, dynamic environments such as MMA bouts.

Once the pose estimator was initialized on the first frame, it was iteratively applied to the remaining frames of the clip. This iterative application allowed the pose estimator to follow the movements of the first fighter consistently, capturing the x and y coordinates of key anatomical landmarks throughout the clip.

3) Landmark Extraction for the Second Fighter

Following the completion of landmark extraction for the first fighter, the process was repeated for the second fighter. The video clip was rewound to the first frame, and the second bounding box was used to mask the image, isolating the second fighter.

The MediaPipe Pose Estimator was reinitialized to focus on this new region, ensuring that the second fighter's landmarks were accurately captured. However, challenges arose when the pose estimations from the first run began to overlap or drift into the second fighter's region. To address this, the SSD MobileNet model was rerun to detect the fighters again, and the bounding boxes were reassessed. The box with the least overlap with the first set of landmarks was selected, ensuring the distinction between the two fighters was maintained.

This rerun and reassessment process was crucial in mitigating the effects of overlapping detections, particularly in scenarios where fighters were in close proximity or engaged in grappling.

4) Iterative Refinement and Landmark Verification

The iterative landmark extraction process continued with the MediaPipe Pose Estimator performing continuous checks for overlap or similarity between the two sets of landmarks. Whenever the landmarks from the second estimation set closely resembled those from the first, the SSD MobileNet model was invoked again, and the bounding boxes were reassessed.

This iterative refinement was essential in maintaining the accuracy and integrity of the landmark data, particularly in the dynamic and unpredictable environment of MMA fighting. The frequent reassessment of bounding boxes helped prevent the model from confusing the two fighters, ensuring that the landmark data remained distinct and reliable.

5) Final Landmark Extraction and Data Compilation

After completing the landmark extraction for both fighters across all frames of a clip, the data was compiled into a structured format. This format included the x and y coordinates for 12 key anatomical landmarks: left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle, and right ankle.

The same process was repeated for all video clips, ensuring consistency in data extraction across the entire dataset. This comprehensive dataset, consisting of detailed pose landmarks for each fighter, provided the foundation for subsequent feature engineering and classification tasks. The extracted landmarks offer a rich representation of the fighters' movements, capturing the spatial and temporal dynamics essential for understanding and classifying different combat moves.

This structured data serves as the basis for advanced feature engineering, where raw positional data points are transformed into more complex and informative features for predictive modeling and classification tasks in MMA movement analysis.

Extracted Data

The resulting extracted data consists of 12 key anatomical landmarks for each fighter. These landmarks are: 1) Left Shoulder, 2) Right Shoulder, 3) Left Elbow, 4) Right Elbow, 5) Left Wrist, 6) Right Wrist, 7) Left Hip, 8) Right Hip, 9) Left Knee, 10) Right Knee, 11) Left Ankle, and 12) Right Ankle.

For each landmark, the x and y coordinates were recorded, providing the pixel positions of these keypoints within each frame. The x component represents the horizontal position, while the y component represents the vertical position in the frame.

This data was compiled into a structured format for further analysis and classification.

Feature Engineering Process

Following the extraction of keypoints from each frame, feature engineering was performed to enhance the representation of the fighters' movements and postures, thereby facilitating more

effective classification of MMA techniques. The process involved calculating distances and angles between the extracted landmarks to derive a comprehensive set of features. Below are the steps and methodologies applied:

Distance Calculation: To quantify the spatial relationships between the fighters, distances between corresponding landmarks of the two individuals were calculated. This included:

- X Component: The horizontal distance between corresponding landmarks.
- Y Component: The vertical distance between corresponding landmarks.

These distance features were designed to capture the spatial dynamics between the fighters. For instance, a significant reduction in the distance between a fighter's wrist or ankle and their opponent, while other landmarks remain relatively stable, might indicate a punch or kick. Similarly, the reduction in distance between the fighters' hips and shoulders, particularly if combined with a lowering of these landmarks relative to the opponent, could suggest a takedown attempt.

Angle Calculation: Angles were computed for specific triplets of keypoints to capture the orientation and movement dynamics of different joints. The following angles were calculated:

- Elbow Angles: Between "left_shoulder," "left_elbow," and "left_wrist"; and between "right_shoulder," "right_elbow," and "right_wrist."
- Shoulder Angles: Between left_hip," "left_shoulder," and "left_elbow,"; and between "right_hip," "right_shoulder," "right_elbow."
- Hip Angles: Between "left_shoulder," "left_hip," and "left_knee"; and between "right_shoulder," "right_hip," and "right_knee."
- Knee Angles: Between "left_hip," "left_knee," and "left_ankle"; and between "right_hip," "right_knee," and "right_ankle."

The rationale for extracting angle features stems from the notion that certain joint configurations are indicative of specific movements. For example, a rapid increase in the angle of the elbow and shoulder may signal a punch, while a quick decrease in the angle of the hip may indicate a kick. These angles were derived using basic trigonometric functions to accurately represent the joint configurations and limb orientations. The angles provide insights into the fighters' postures and movement patterns, which are pivotal for distinguishing between different actions in MMA.

Feature Variables: The engineered features for each frame include:

- Landmark Coordinates: X and Y coordinates for each keypoint of both fighters.
- Distance Features: Distances between corresponding keypoints of the two fighters, represented as _dist_x and _dist_y.

- Angle Features: Angles between specific triplets of keypoints, representing joint and limb orientations.

The comprehensive list of feature variables contained the X and Y position values for each fighter's left and right shoulders, elbows, wrists, hips, knees, and ankles. Additionally, the features contained the X and Y distances between the anatomical landmarks of each fighter relative to their opponent, as well as the angles of each of their left and right shoulder, elbow, hip, and knee joints. These engineered features, encompassing both spatial and angular information, provide a detailed representation of the fighters' movements and postures.

The total number of pose features = (number of keypoints)*(number of Cartesian components)*(number of fighters) + (number of joint angles)*(number of fighters) + (number of distances between each fighter and their opponent's keypoints)*(number of Cartesian coordinates) = 12*2*2 + 8*2 + 12*2 = 88 features (90 total features including the time component and video clip identifier.)

(Please note: due to a coding oversight on my end, angle features were only calculated for single fighter, and due to time constraints, I was not able to correct this before the due date. Due to the hardware limitations of my laptop, the extractor takes half a day to extract features from a video. Thus, for the rest of the project, ***the total number of pose features = 80, 82 including the time component and video clip identifier****.)*

To further enhance the utility of these features, all data were normalized using a standard scaler, which is a common practice in neural networks to ensure that features contribute equally to the model's learning process. The rationale for normalization is grounded in the need to maintain a consistent scale across features, preventing any single feature from disproportionately influencing the model's predictions.

The engineered features, informed by domain knowledge, are designed to capture the nuanced dynamics of MMA techniques. This enriched feature set supports robust analysis and classification of MMA techniques based on the fighters' actions and positions.

Data Preparation

The dataset consisted of 80 pose estimation features extracted and engineered from video clips, with each frame labeled according to the specific action being performed as well as 2 features indicating which video clip the frames came from and what their time index in that clip was. The initial step in data preparation involved cleaning the dataset by removing duplicate entries, resetting the index, and addressing missing values. Missing values were filled using data from the preceding frame, a technique that may reduce the negative impact of missing data on

model performance. This preprocessing step ensured the consistency and completeness of the input data, which is crucial for the reliability of the predictive models.

For classification purposes, the action labels were consolidated into five categories: "ground," "kick," "not_engaged," "punch," and "takedown." Given the potential for class imbalances, particularly in video datasets where certain actions may be underrepresented, I employed the RandomUnderSampler technique. This method balances the training set by undersampling the majority classes, thereby preventing the model from becoming biased towards more frequent actions (Liu et al., 2009). This step was essential to ensure that the model's performance was not skewed by the prevalence of specific classes in the training data.

The time component and clip source identifier was removed for the FNN training set but kept for the LSTM training set. To prepare the data for the LSTM, the data was grouped by their source video clip and converted into a 3-dimentional array with the dimensions being: pose features (columns), frames (rows), and sequence number (time component).

Predictive Modeling

To classify MMA techniques based on extracted pose features, I developed two distinct models: a Feedforward Neural Network (FNN) and a Long Short-Term Memory (LSTM) network. These models represent different approaches to handling the input data: the FNN processes each video frame as an isolated instance, while the LSTM considers sequences of frames, capturing the temporal dependencies between them.

Feedforward Neural Network (FNN)

Feedforward Neural Networks (FNNs) are a foundational architecture in machine learning, particularly in image and video analysis tasks. An FNN consists of an input layer, one or more hidden layers, and an output layer, where data flows in one direction from input to output without any feedback loops (Goodfellow et al., 2016). FNNs are effective for tasks where the assumption of independence between input features holds, making them suitable for baseline classification models when temporal information is not a primary concern. However, they may struggle with tasks requiring the integration of contextual or sequential information, which is where recurrent models like LSTMs become advantageous.

As a baseline model, I implemented a Feedforward Neural Network (FNN) to classify the actions based on the pose estimation features. The input to the FNN consisted of flattened feature vectors for each frame, effectively treating each frame as an independent instance without any consideration of temporal context. The architecture of the FNN included three fully connected

dense layers, each followed by a ReLU activation function to introduce non-linearity. Dropout layers were incorporated after each dense layer to mitigate the risk of overfitting by randomly deactivating a fraction of the neurons during training.

The output layer of the FNN was a softmax layer, which provided probability distributions across the five action classes. The model was compiled using the Adam optimizer, known for its efficiency and adaptability in training neural networks, with categorical cross-entropy as the loss function. The primary evaluation metric was accuracy, reflecting the proportion of correctly classified instances. The model was trained for 169 epochs with a batch size of 16, a configuration that was empirically determined to balance computational efficiency with convergence.

The FNN serves as a standard benchmark in this study, providing a reference point for evaluating the added value of incorporating sequential information through the LSTM model. While FNNs are effective in many classification tasks, their lack of temporal awareness makes them less suited for tasks where the sequence of inputs carries significant meaning, such as in video-based action recognition.

Long Short-Term Memory (LSTM) Network

Long Short-Term Memory (LSTM) networks, a specific type of Recurrent Neural Network (RNN), are designed to capture long-range dependencies in sequential data. Unlike traditional RNNs, LSTMs incorporate mechanisms to mitigate the vanishing gradient problem, allowing them to maintain a memory of previous inputs over longer sequences (Hochreiter & Schmidhuber, 1997). This ability makes LSTMs particularly effective in tasks involving time-series data or sequences of images, such as video analysis, where understanding the temporal progression of frames is critical. LSTMs have been successfully applied in various domains, including human activity recognition in sports, where they outperform non-sequential models by leveraging temporal dependencies (Ullah et al., 2021).

To capture the temporal structure inherent in video data, I developed a Long Short-Term Memory (LSTM) network. LSTMs are uniquely suited for sequential data due to their ability to maintain and update a memory cell that tracks dependencies over time, enabling them to model long-range relationships between inputs (Hochreiter & Schmidhuber, 1997). The input to the LSTM model was a sequence of feature vectors, each representing a frame within a sliding window of 16 consecutive frames. This window size was selected to balance the need for capturing short-term dependencies with the computational demands of processing longer sequences.

The architecture of the LSTM model included a Bidirectional LSTM layer, which allows the model to consider both past and future frames within the sequence, effectively enhancing its understanding of temporal context. Following the LSTM layer, a dropout layer was added to prevent overfitting. A masking layer was also included to handle sequences of varying lengths, a common issue in real-world video data where sequences may not be perfectly aligned. The output layer of the model was a dense layer with a softmax activation function, similar to the FNN, providing a probability distribution over the five classes.

The choice of an LSTM was driven by its proven efficacy in handling sequential data, as demonstrated in previous studies on human activity recognition (Noori et al., 2019; Cob-Parro et al., 2024). By leveraging the temporal relationships between frames, the LSTM was expected to outperform the FNN, particularly for actions that unfold over multiple frames and require an understanding of the sequence of movements.

Evaluation Metrics

To assess the performance of both models, I used a held-out test set and evaluated them primarily on accuracy. Additionally, precision, recall, and F1-score were calculated to provide a more nuanced understanding of the models' performance across the different action classes (Sokolova & Lapalme, 2009). These metrics were crucial for determining whether the LSTM's ability to capture temporal dependencies translated into superior classification accuracy compared to the FNN, particularly for complex actions that require an understanding of the sequence of movements.

Results

Overview

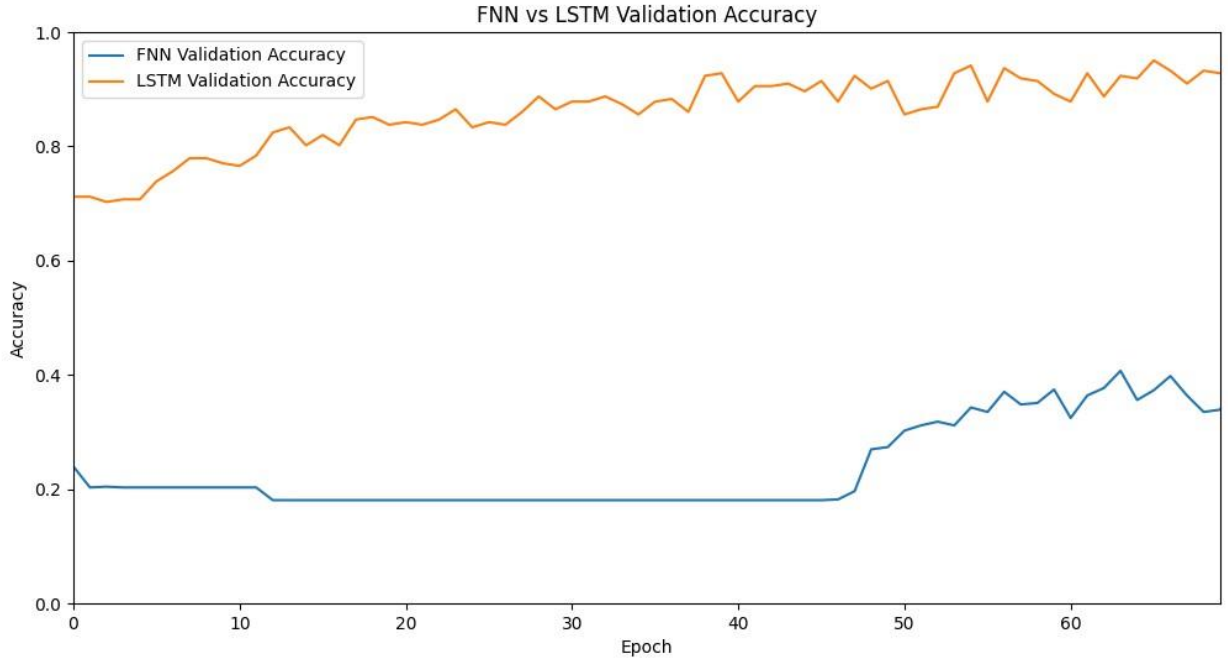
The performance of the Feedforward Neural Network (FNN) and Long Short-Term Memory (LSTM) models in classifying MMA techniques was evaluated using standard classification metrics: precision, recall, F1-score, and accuracy. The results demonstrate that the LSTM model significantly outperformed the FNN in almost all metrics validating the hypothesis that incorporating temporal information leads to better classification of dynamic actions in video data. The results underscore the importance of model architecture in handling sequential data, particularly in applications like MMA technique classification, where the temporal evolution of actions is key to accurate prediction.

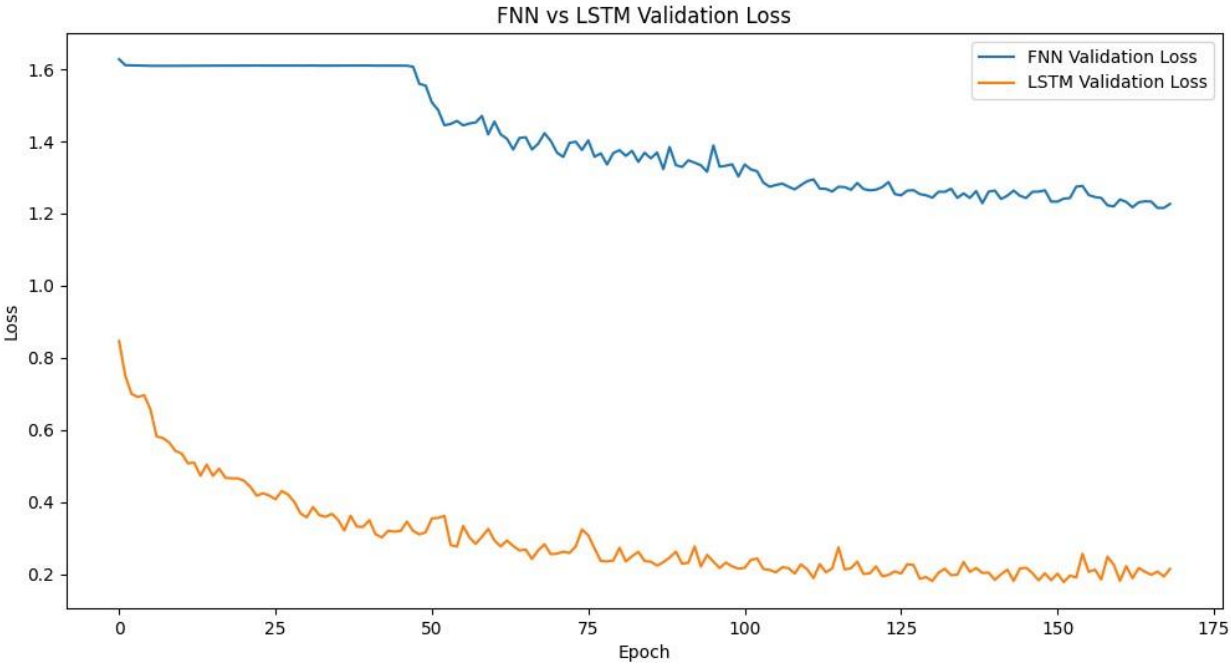
Classification Reports

FNN	Precision	Recall	F1-score	Support
ground	0.89	0.46	0.61	104
kick	0	0	0	93
not_engaged	0.22	0.19	0.2	80
punch	0.3	0.83	0.45	69
takedown	0.47	0.7	0.56	79
accuracy			0.41	425
macro avg	0.38	0.43	0.36	425
weighted avg	0.4	0.41	0.36	425

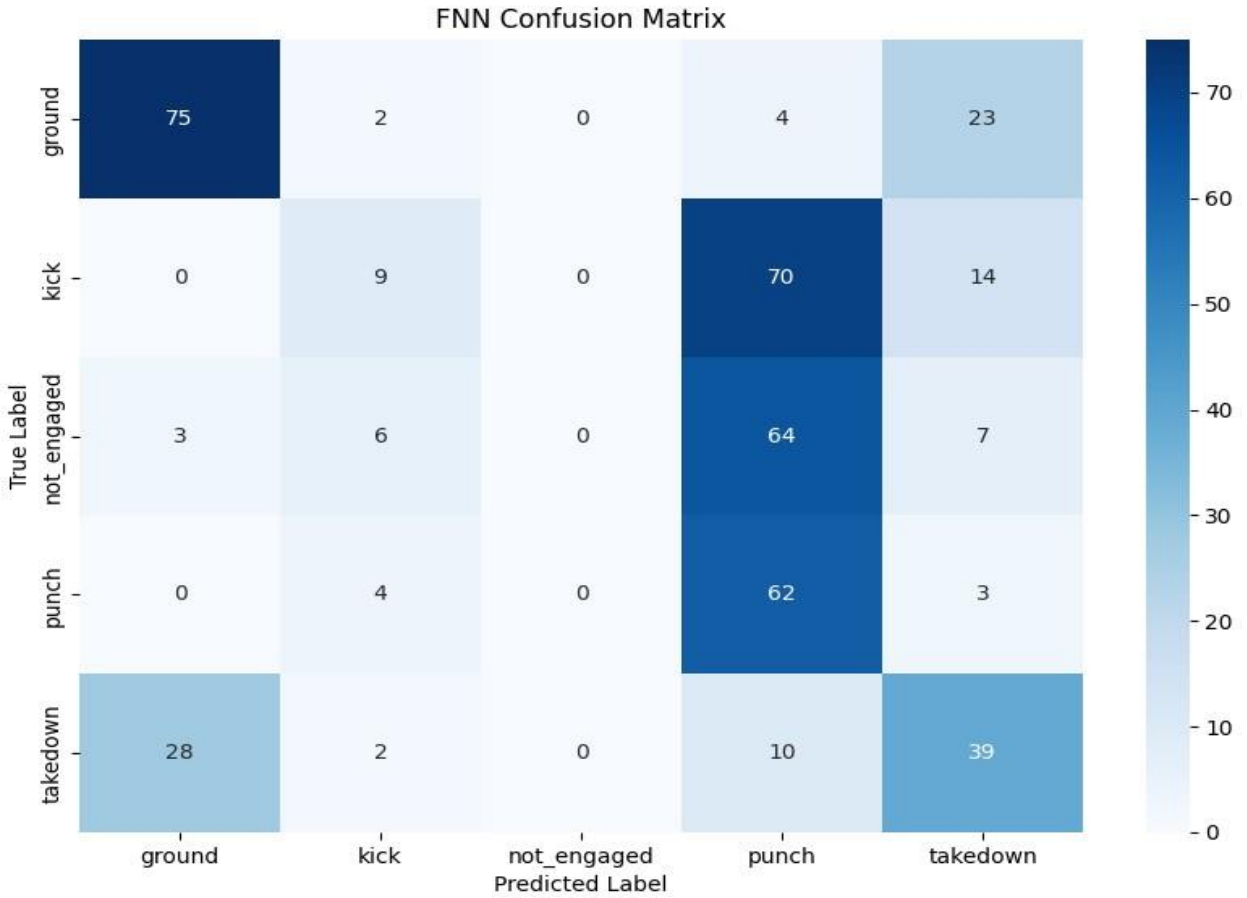
LSTM	Precision	Recall	F1-score	Support
ground	1	0.46	0.61	104
kick	1	0.46	0.63	13
not_engaged	0.8	0.95	0.87	21
punch	0.83	0.83	0.83	6
takedown	0.94	1	0.97	48
accuracy			0.93	124
macro avg	0.91	0.85	0.86	124
weighted avg	0.94	0.93	0.92	124

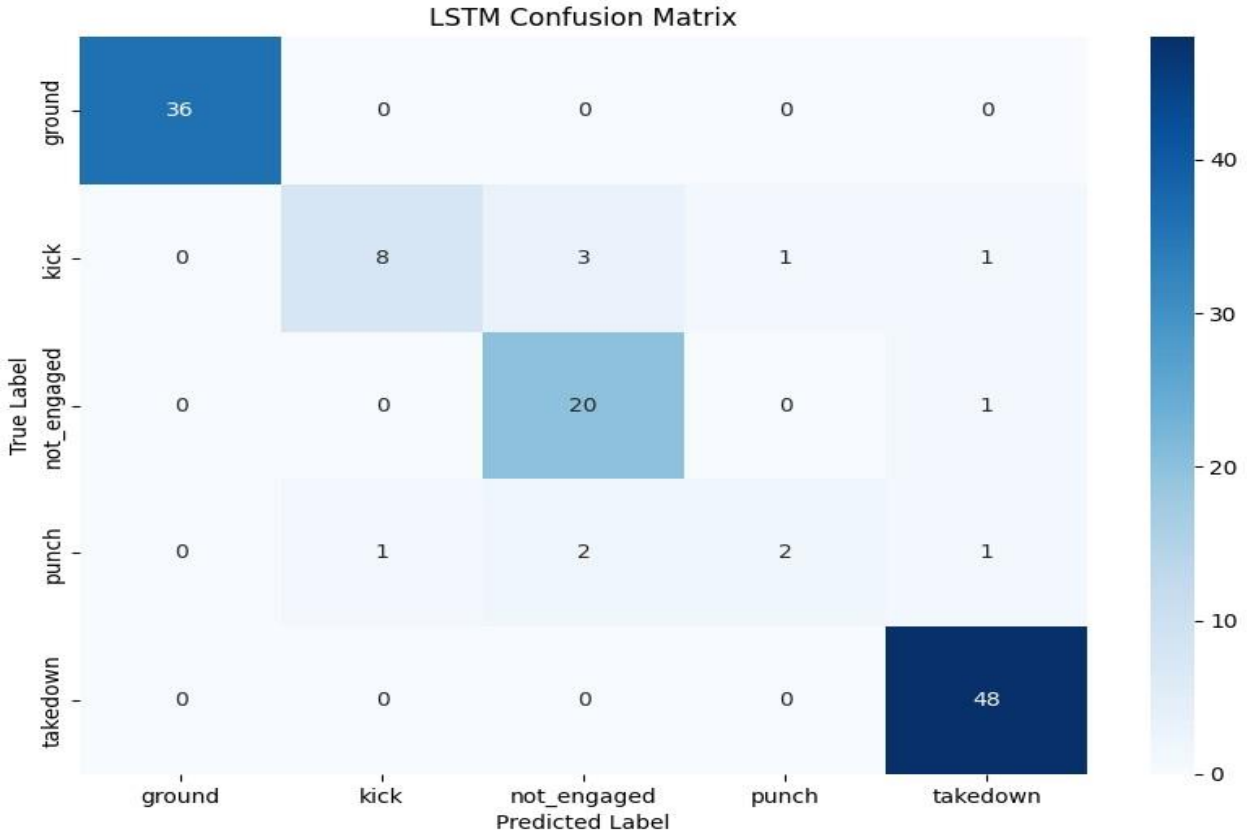
Validation Accuracy and Loss Plots





Confusion Matrices





Feedforward Neural Network (FNN)

The FNN served as a baseline model, where each frame of the video was treated as an independent instance. The model achieved an overall accuracy of 41%, with notable class imbalances affecting performance. For instance, the model was unable to correctly classify the "kick" action, as indicated by a precision and recall of 0. This suggests that the FNN struggled to distinguish "kick" from other actions, likely due to the lack of temporal context in the data. The F1-score for "ground" was 0.61, while for "punch," the recall was surprisingly high at 0.83, but precision was low at 0.30, resulting in a moderate F1-score of 0.45. This disparity indicates that while the FNN could detect "punch" actions frequently, it also misclassified other actions as "punch," leading to lower precision. It also appears that the FNN is clearly able to discern the difference between standing positions (‘punch’, ‘kick’, and ‘not_engaged’) from grappling positions (‘takedown’ and ‘ground’), but unable to discern differences within standing and grappling positions.

Long Short-Term Memory (LSTM) Network

The LSTM model, designed to capture temporal dependencies in sequential data, achieved a much higher accuracy of 93%. This improvement is particularly evident in the classification of "takedown" and "not_engaged" actions, with F1-scores of 0.97 and 0.87, respectively. The "kick" class, which the FNN failed to classify correctly, saw a significant improvement with an F1-score of 0.63, although the recall remained at 0.46, suggesting that the LSTM could identify "kick" actions but still missed some instances. The LSTM's ability to maintain a memory of previous frames likely contributed to this performance boost, enabling it to more accurately distinguish between similar actions that unfold over multiple frames.

Model Comparison

The overall performance metrics—precision, recall, and F1-score—are consistently higher for the LSTM model across all classes. The macro average F1-score for the LSTM was 0.86, compared to 0.36 for the FNN, underscoring the LSTM's superior ability to classify actions in sequential video data. The improvement is most pronounced in the "takedown" and "not_engaged" classes, where the temporal dynamics of the actions are critical for accurate classification.

Error Analysis

The FNN's poor accuracy in the "kick" classes and poor precision with “punch”, “kick”, and “not_engaged” classes and its general tendency to misclassify frames suggest that a simple feedforward architecture is insufficient for capturing the complex, dynamic nature of MMA techniques. In contrast, the LSTM's strong performance across most classes can be attributed to its recurrent structure, which allows it to account for the temporal progression of actions.

Conclusion

The LSTM model significantly outperforms the FNN, validating the hypothesis that incorporating temporal information leads to better classification of dynamic actions in video data. The results underscore the importance of model architecture in handling sequential data, particularly in applications like MMA technique classification, where the temporal evolution of actions is key to accurate prediction.

Discussion

Summary of the Findings

This study aimed to develop and evaluate a methodology for classifying Mixed Martial Arts (MMA) techniques using advanced pose estimation and machine learning techniques. By systematically capturing and analyzing fighter movements within a controlled environment, the research demonstrated the feasibility of using an object detector to adapt a pose estimator for combat sports video data and using models that leverage sequence data for accurately identifying and categorizing various combat actions. The results offer valuable insights into the application of pose estimation for sports analysis, particularly in dynamic and unpredictable settings like MMA.

The methodology employed in this study involved several critical steps: data collection, video preparation and labeling, pose estimation and landmark extraction, feature engineering, and predictive modeling. The data collection process ensured comprehensive coverage of the fighters' movements, with high-definition cameras strategically placed around the UFC Gym Octagon. The manual labeling of video clips, though labor-intensive, allowed for precise categorization of combat actions into five predefined labels: Punch, Kick, Takedown, Ground, and Not Engaged.

Pose estimation and landmark extraction were central to this study, providing detailed data on fighters' anatomical positions across frames. By calculating distances and angles between key landmarks, the study derived a set of features that effectively captured the spatial and temporal dynamics of the fighters' movements. Theoretically, this would result in 88 (89 when including the time component) training features. However, due to a coding error, the total number of training features was 80 (81). These features were then used to train two machine learning models: a Feedforward Neural Network (FNN) and a Long Short-Term Memory (LSTM) network.

The FNN model, which treated each frame independently, demonstrated the potential for classifying MMA techniques based on static spatial data. However, the LSTM model, which incorporated temporal dependencies by processing sequences of frames, achieved superior performance. This finding highlights the importance of considering the temporal progression of movements when analyzing complex actions like those in MMA.

Relevance

The results of this study underscore the effectiveness of combining pose estimation with object detection as a method for extracting data from MMA videos for use in training predictive models. This study shows that the pipelines previously suggested by Cob-Parro et al. (2023) and Noori et al. (2019) for human action recognition can be adapted and applied to combat sports. The LSTM model's success in capturing temporal dependencies suggests that future research should continue exploring sequence-based models for sports analysis. Moreover, the detailed feature engineering process, which included distance and angle calculations, proved essential for distinguishing between different types of movements. These findings support the broader application of pose estimation in sports, where understanding the intricate dynamics of athletes' movements is crucial.

While the methodology developed in this study showed promising results, several limitations were identified. The manual labeling process, conducted by a single labeler, may introduce bias and reduce the generalizability of the findings. Additionally, the study's focus on a controlled environment, with a specific camera setup, may limit its applicability to other settings or sports. Future research should address these limitations by incorporating multiple labelers to enhance the robustness of the dataset and exploring different camera configurations to test the methodology in more varied environments.

Moreover, the data collection environment could have been more controlled. The data used in this study was recorded at a local public gym with older phone cameras, which likely affected the data quality. Although I have amateur fighting experience, the recorded techniques would have benefited from the oversight of an MMA domain expert, ensuring more relevant and accurate video data. These aspects underline the potential for improvements in the data collection process in future studies.

Future Directions

Building on the findings of this study, several avenues for future research can be pursued:

1. Enhanced Data Collection: Future iterations should prioritize data collection in a more controlled environment with high-definition cameras and consistent lighting. Additionally, involving an MMA domain expert during the recording process could ensure the data captures the most relevant and technically accurate movements.
2. Improved Video Data Representation: The data extraction process, which utilized the SSD MobileNet model for people detection and the MediaPipe Pose Estimator for

landmark extraction, could be refined by exploring alternative methods. Techniques such as optical flow could help reduce background noise, while other pose estimators or approaches like silhouetting might offer more accurate representations of the fighters.

3. Refinement of Pose Estimation Techniques: While the MediaPipe Pose Estimator was effective in capturing landmarks, future studies could explore more advanced pose estimation models that offer higher accuracy and robustness in complex scenarios, such as when fighters are in close proximity or grappling.
4. Exploration of Advanced Machine Learning Models: Beyond the LSTM model, future research could explore the use of Generative Adversarial Networks (GANs), Transformer networks, or other advanced algorithms that handle long-range dependencies and complex temporal patterns more effectively. Building on transfer learning with models like AlexNet or VGG19, or applying specialized video models such as UCF101, could also enhance the model's performance.
5. Application to Other Sports: The methodology developed in this study could be adapted and tested in other sports where pose estimation and movement analysis are crucial, such as gymnastics, basketball, or soccer. Comparative studies across different sports could reveal the strengths and limitations of the approach in various contexts.
6. Generalization and Marketability: By implementing these enhancements, the resulting model could become more generalizable to MMA bouts outside the specific location where data was collected. This broader applicability would increase its potential as a marketable tool for MMA coaches and fight promotions, offering real-world value beyond academic research.
7. Real-Time Application and Feedback: Future research could focus on the real-time application of this methodology, enabling live analysis and feedback during training sessions or competitions. This would require optimization of the pose estimation and classification processes to operate efficiently in real-time environments.

In conclusion, this study demonstrates the potential of using pose estimation and machine learning to classify MMA techniques, offering a foundation for future research in sports analytics. By addressing the identified limitations and exploring new directions, researchers can continue to advance the field, contributing to the development of more sophisticated and applicable models for understanding athletic performance.

References

For your convenience, I have assembled all references in pdf form here: https://drive.google.com/drive/folders/1CSmEz_dzm5wVeaptGhJmCa4eIbehARty?usp=drive_link

Cob-Parro, A. C., Losada-Gutiérrez, C., Marrón-Romera, M., Gardel-Vicente, A., & Bravo-Muñoz, I. (2023). A new framework for deep learning video-based human action recognition on the edge. Expert Systems with Applications, 238(Part E), 122220. https://doi.org/10.1016/j.eswa.2023.122220

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. https://www.deeplearningbook.org/contents/mlp.html

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Kasiri, S., Fookes, C., Morgan, S., Martin, D., & Sridharan, S. (2015). Combat sports analytics: Boxing punch classification using overhead depth imagery. In Proceedings of the IEEE International Conference on Image Processing (ICIP), 4545-4549. https://doi.org/10.1109/ICIP.2015.7351667

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision (ECCV), 21-37.

Liu, Y., Wu, X., & Zhou, Z.-H. (2009). Exploratory under-sampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 39(2), 539-550. https://doi.org/10.1109/TSMCB.2008.2007853

Lugaresi, C., Tang, J., Nash, H., McLeod, K., Hua, F., Chang, C., ... & Roncal, C. (2019). MediaPipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172.

Minhas, R. A., Javed, A., Irtaza, A., Mahmood, M. T., & Joo, Y. B. (2019). Shot classification of field sports videos using AlexNet convolutional neural network. Applied Sciences, 9(3), 483. https://doi.org/10.3390/app9030483

Noori, F. M., Wallace, B., Uddin, M. Z., & Torresen, J. (2019). A robust human activity recognition approach using OpenPose, motion features, and deep recurrent neural network. In M. Felsberg,

P. E. Forssén, I. M. Sintorn, & J. Unger (Eds.), Image analysis (pp. 317-328). Springer, Cham. https://doi.org/10.1007/978-3-030-20205-7_25

Podgorelec, V., Pečnik, Š., & Vrbančič, G. (2020). Classification of similar sports images using convolutional neural network with hyper-parameter optimization. Applied Sciences, 10(23), 8494. https://doi.org/10.3390/app10238494

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., ... & Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1297-1304). IEEE. https://doi.org/10.1109/CVPR.2011.5995316

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427-437. https://doi.org/10.1016/j.ipm.2009.03.002

Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2018). Action recognition in video sequences using deep bi-directional LSTM with CNN features. IEEE Access, 6, 1155-1166. https://doi.org/10.1109/ACCESS.2017.2778011

Ullah, M., Yamin, M.M., Mohammed, A.K., Khan, S.D., Ullah, H., & Cheikh, F.A. (2021). Attention-based LSTM Network for Action Recognition in Sports. IRIACV.

Ye, M., Zhang, Q., Wang, L., Zhu, J., Yang, R., & Gall, J. (2013). A survey on human motion analysis from depth data. In M. Grzegorzek, C. Theobalt, R. Koch, & A. Kolb (Eds.), Time-of-flight and depth imaging. Sensors, algorithms, and applications (Vol. 8200, pp. 149-182). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-44964-2_8

Appendix

The ipynb files, collected dataset, object detection models used in the study, and some unused additional visualizations can be found in the appendix folder of my Google Drive here: https://drive.google.com/drive/folders/1CSmEz_dzm5wVeaptGhJmCa4eIbehARty?usp=drive_link

The following is the code used for feature extraction, engineering, labeling, data preparation, model training, and model evaluation:

```
# Functions used in the feature extraction and feature engineering process
import cv2
import mediapipe as mp
import tensorflow as tf
import numpy as np
import csv
import math

# Load SSD MobileNet model
model_ssd = tf.saved_model.load('ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model')

# Initialize MediaPipe Pose model
mp_pose = mp.solutions.pose

necessary_landmark_indices = [11, 12, 13, 14, 15, 16, 23, 24, 25, 26, 27, 28]

keypoint_names = [
    "left_shoulder", "right_shoulder", "left_elbow", "right_elbow",
    "left_wrist", "right_wrist", "left_hip", "right_hip",
    "left_knee", "right_knee", "left_ankle", "right_ankle"
]

angle_connections = [
    ("left_shoulder", "left_elbow", "left_wrist"),
    ("right_shoulder", "right_elbow", "right_wrist"),
    ("left_hip", "left_shoulder", "left_elbow"),
    ("right_hip", "right_shoulder", "right_elbow"),
    ("left_hip", "left_knee", "left_ankle"),
    ("right_hip", "right_knee", "right_ankle"),
    ("left_shoulder", "left_hip", "left_knee"),
    ("right_shoulder", "right_hip", "right_knee")
]

def non_max_suppression(boxes, scores, iou_threshold=0.5):
    indices = tf.image.non_max_suppression(
        boxes=boxes,
        scores=scores,
        iou_threshold=iou_threshold,
        max_output_size=2,
        score_threshold=0.2
    )
    selected_boxes = tf.gather(boxes, indices).numpy()
    return selected_boxes

def detect_people(frame, max_boxes=2):
    resized_frame = cv2.resize(frame, (320, 320))
    input_tensor = tf.convert_to_tensor(cv2.cvtColor(resized_frame, cv2.COLOR_BGR2RGB))
    input_tensor = input_tensor[tf.newaxis, ...]

    detections = model_ssd(input_tensor)
    boxes = detections['detection_boxes'][0].numpy()
```

```
    scores = detections['detection_scores'][0].numpy()
    classes = detections['detection_classes'][0].numpy()

    person_boxes = [box for box, score, cls in zip(boxes, scores, classes) if score > 0.2 and cls == 1]
    person_scores = [score for score, cls in zip(scores, classes) if score > 0.2 and cls == 1]

    if not person_boxes:
        return []

    selected_boxes = non_max_suppression(np.array(person_boxes), np.array(person_scores))

    largest_boxes = sorted(selected_boxes, key=lambda box: (box[2] - box[0]) * (box[3] - box[1]), reverse=True)[:max_boxes]

    return largest_boxes

def mask_image_with_box(frame, box):
    mask = np.zeros_like(frame)
    h, w, _ = frame.shape
    y_min, x_min, y_max, x_max = box
    mask[int(y_min * h):int(y_max * h), int(x_min * w):int(x_max * w)] = frame[int(y_min * h):int(y_max * h), int(x_min * w):int(x_max * w)]
    return mask

def estimate_pose(image):
    with mp_pose.Pose(static_image_mode=False, model_complexity=0, min_detection_confidence=0.2) as pose:
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        result = pose.process(image_rgb)
        return result.pose_landmarks

def filter_landmarks(landmarks, indices):
    if landmarks:
        return [landmarks.landmark[idx] for idx in indices]
    return [None] * len(indices)

def are_poses_similar(pose1, pose2, threshold=0.1):
    if not pose1 or not pose2:
        print(f"Invalid poses: pose1={pose1}, pose2={pose2}")
        return False  # Or handle as appropriate

    similar_count = 0
    for l1, l2 in zip(pose1, pose2):
        if l1 is None or l2 is None:
            print(f"Skipping None value: l1={l1}, l2={l2}")
            continue
        distance = np.sqrt((l1.x - l2.x) ** 2 + (l1.y - l2.y) ** 2)
        if distance < threshold:
            similar_count += 1
    return similar_count

def calculate_angle(p1, p2, p3):
    def vector_from_points(p1, p2):
        return (p2[0] - p1[0], p2[1] - p1[1])

    def dot_product(v1, v2):
        return v1[0] * v2[0] + v1[1] * v2[1]

    def magnitude(v):
        return math.sqrt(v[0]**2 + v[1]**2)

    v1 = vector_from_points(p2, p1)
    v2 = vector_from_points(p2, p3)

    dot_prod = dot_product(v1, v2)
    mag_v1 = magnitude(v1)
    mag_v2 = magnitude(v2)

    if mag_v1 == 0 or mag_v2 == 0:
```

```
        return 0.0

    cos_theta = dot_prod / (mag_v1 * mag_v2)
    angle = math.degrees(math.acos(min(max(cos_theta, -1.0), 1.0)))

    return angle

def process_video_for_landmarks_csv(input_video_path, output_csv_path, starting_frame, label):
    cap = cv2.VideoCapture(input_video_path)

    if not cap.isOpened():
        print("Error: Unable to open the video.")
        return

    # Initialize storage for pose estimations
    all_pose_estimations = [[], []]

    # Step 1: Detect people in the first frame
    ret, first_frame = cap.read()
    if not ret:
        print("Error: Unable to read the video.")
        return

    boxes = detect_people(first_frame)
    if len(boxes) < 2:
        print("Error: Less than 2 people detected in the first frame.")
        return

    # Step 2: Initialize pose model and estimate pose for first person in first frame
    masked_frame = mask_image_with_box(first_frame, boxes[0])
    initial_landmarks = estimate_pose(masked_frame)
    filtered_landmarks = filter_landmarks(initial_landmarks, necessary_landmark_indices)
    all_pose_estimations[0].append(filtered_landmarks)

    # Process the rest of the video for the first person
    frame_idx = 1
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        pose_landmarks = estimate_pose(frame)
        filtered_landmarks = filter_landmarks(pose_landmarks, necessary_landmark_indices)
        all_pose_estimations[0].append(filtered_landmarks)

        frame_idx += 1

    # Step 3: Rewind, initialize pose model, and estimate pose for second person in first frame
    cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
    masked_frame = mask_image_with_box(first_frame, boxes[1])
    initial_landmarks = estimate_pose(masked_frame)
    filtered_landmarks = filter_landmarks(initial_landmarks, necessary_landmark_indices)
    all_pose_estimations[1].append(filtered_landmarks)

    # Process the rest of the video for the second person with overlap detection
    frame_idx = 1
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        pose_landmarks = estimate_pose(frame)
        filtered_landmarks = filter_landmarks(pose_landmarks, necessary_landmark_indices)

        # Check for overlap with the first person's pose in the same frame
        if frame_idx < len(all_pose_estimations[0]) and are_poses_similar(filtered_landmarks, all_pose_estimations[0][frame_idx]):
```

```
            # Re-detect people and adjust
            boxes = detect_people(frame)
            if len(boxes) > 1:
                centers = [((box[1] + box[3]) / 2, (box[0] + box[2]) / 2) for box in boxes]
                est_center = (np.mean([lm.x for lm in filtered_landmarks]), np.mean([lm.y for lm in filtered_landmarks]))
                distances = [np.linalg.norm(np.array(center) - np.array(est_center)) for center in centers]
                furthest_box_idx = np.argmax(distances)
                masked_frame = mask_image_with_box(frame, boxes[furthest_box_idx])
                pose_landmarks = estimate_pose(masked_frame)
                filtered_landmarks = filter_landmarks(pose_landmarks, necessary_landmark_indices)

        all_pose_estimations[1].append(filtered_landmarks)
        frame_idx += 1

    cap.release()

    # Save the landmarks and differences to a CSV file
    with open(output_csv_path, 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile)

        # Write header
        header = ['Frame']
        for keypoint in keypoint_names:
            header.extend([f"{keypoint}_x1", f"{keypoint}_y1", f"{keypoint}_x2", f"{keypoint}_y2"])
            header.extend([f"{keypoint}_dist_x", f"{keypoint}_dist_y"])
        for (kp1, kp2, kp3) in angle_connections:
            header.append(f"{kp1}_{kp2}_{kp3}_angle")
        header.append('Starting_Frame')
        header.append('Label')
        csvwriter.writerow(header)

        # Write pose data
        frame_idx = 0
        while frame_idx < len(all_pose_estimations[0]) or frame_idx < len(all_pose_estimations[1]):
            row = [frame_idx]
            landmarks1 = all_pose_estimations[0][frame_idx] if frame_idx < len(all_pose_estimations[0]) else [None] *
len(necessary_landmark_indices)
            landmarks2 = all_pose_estimations[1][frame_idx] if frame_idx < len(all_pose_estimations[1]) else [None] *
len(necessary_landmark_indices)

            for i in range(len(necessary_landmark_indices)):
                if landmarks1[i]:
                    row.extend([landmarks1[i].x, landmarks1[i].y])
                else:
                    row.extend([None, None])

                if landmarks2[i]:
                    row.extend([landmarks2[i].x, landmarks2[i].y])
                else:
                    row.extend([None, None])

                if landmarks1[i] and landmarks2[i]:
                    row.extend([landmarks2[i].x - landmarks1[i].x, landmarks2[i].y - landmarks1[i].y])
                else:
                    row.extend([None, None])

            for (kp1, kp2, kp3) in angle_connections:
                idx1, idx2, idx3 = keypoint_names.index(kp1), keypoint_names.index(kp2), keypoint_names.index(kp3)
                if landmarks1[idx1] and landmarks1[idx2] and landmarks1[idx3]:
                    angle = calculate_angle(
                        (landmarks1[idx1].x, landmarks1[idx1].y),
                        (landmarks1[idx2].x, landmarks1[idx2].y),
                        (landmarks1[idx3].x, landmarks1[idx3].y)
                    )
                    row.append(angle)
                else:
```

```
            row.append(None)

        row.append(starting_frame)
        row.append(label)
        csvwriter.writerow(row)
        frame_idx += 1

    print(f"Landmarks and differences saved to {output_csv_path}")
```

```
# Extract, engineer, and label landmark data
import os
import pandas as pd

input_path = 'training_frames/sorted_clips/'
output_path = 'extracted_data/'

for subdir, _, files in os.walk(input_path):
    label = os.path.basename(subdir)

    for file in files:

        if file.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif', '.mp4')):
            in_path = subdir + '/' + file
            out_path = output_path + file.split('.')[0] + '.csv'
            process_video_for_landmarks_csv(in_path, out_path, int(file.split('.')[0].split('_')[1]), label)
```

```
# Prepare extracted data
import pandas as pd
from sklearn.preprocessing import StandardScaler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler

# Load data
df = pd.read_csv('extracted_video_data.csv')
df = df.drop_duplicates()
df = df.reset_index(drop=True)

for index, row in df.iterrows():
    if row.isnull().any():
        df.loc[index] = df.loc[index].fillna(df.loc[index-1])

# Add simpler labels
class_dict = {'closed_guard':'ground', 'open_guard':'ground', 'pin':'ground', 'submission':'ground',
'front_kick':'kick', 'head_kick':'kick', 'leg_kick':'kick', 'mid_kick':'kick',
'body_hook':'punch', 'jab':'punch', 'lead_hook':'punch', 'rear_hook':'punch', 'straight':'punch',
'double_leg':'takedown', 'single_leg':'takedown', 'sprawl':'takedown',
'not_engaged':'not_engaged'}
df['Label_Simplified'] = df['Label'].map(class_dict)

# Convert 'Frame' to an integer or datetime type if it's not already
df['Frame'] = df['Frame'].astype(int)
df = df.drop(columns=['Unnamed: 0', 'Label'])

# Balance the training set using RandomUnderSampler
X = df.drop(columns=['Label_Simplified'])
y = df['Label_Simplified']
rus = RandomUnderSampler(random_state=42)
ros = RandomOverSampler(random_state=42)
X_balanced, y_balanced = rus.fit_resample(X, y)

# Encode y for LSTM
y_dummies = pd.get_dummies(y_balanced)
```

```
df1h = X_balanced.join(y_dummies)
```

```
# Train FNN model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

label_columns = ['ground', 'kick', 'not_engaged', 'punch', 'takedown']

X_flat = np.array(df1h.drop(columns=['Frame', 'Starting_Frame', 'Device', 'ground', 'kick', 'not_engaged', 'punch', 'takedown']))
y_flat = np.array(df1h[label_columns])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_flat, y_flat, test_size=0.1, random_state=42)

# Define the feedforward neural network model
mlp_model = Sequential()
mlp_model.add(Dense(64, input_dim=X_flat.shape[1], activation='relu'))
mlp_model.add(Dropout(0.5))
mlp_model.add(Dense(32, activation='relu'))
mlp_model.add(Dropout(0.5))
mlp_model.add(Dense(16, activation='relu'))
mlp_model.add(Dropout(0.5))
mlp_model.add(Dense(y_flat.shape[1], activation='softmax'))

# Compile the model
optimizer = Adam(learning_rate=0.001)
mlp_model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
mlp_history = mlp_model.fit(X_train, y_train, epochs=169, batch_size=16, validation_split=0.2, verbose=0)

mlp_model.save('mlp_model.h5')
mlp_model = load_model('mlp_model.h5')
```

```
# Evaluate the FNN model on test data
test_loss, test_accuracy = mlp_model.evaluate(X_test, y_test, verbose=0)
print(f'Test Accuracy: {test_accuracy:.2f}')

# Make predictions
y_pred = mlp_model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Classification report
report = classification_report(y_true, y_pred_classes, target_names=label_columns)
print(report)

# Save classification report to a text file
with open('classification_report_mlp.txt', 'w') as f:
    f.write(report)
```

```
# Confusion matrix for FNN
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
# Compute confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred_classes)

# Plot confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_columns, yticklabels=label_columns)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('MLP Confusion Matrix')
plt.savefig('confusion_matrix_mlp.png')
plt.show()
```

```
# Prepare data for LSTM
import numpy as np

# Define the sequence length (window size)
sequence_length = 16  # Adjust this based on your specific needs

# Initialize sequences
X_seq, y_seq = [], []

# Extract label columns
label_columns = ['ground', 'kick', 'not_engaged', 'punch', 'takedown']
grouping_columns = ['Device', 'Starting_Frame']

# Extract feature columns (everything except 'Frame', 'Starting_Frame', 'Device', and labels)
feature_columns = [col for col in df1h.columns if col not in label_columns]
feature_columns = [col for col in feature_columns if col not in grouping_columns]
feature_columns.remove('Frame')

# Sort the dataframe by 'Frame' to ensure the correct order
df_sorted = df1h.sort_values(by=['Device', 'Starting_Frame', 'Frame'])
df_sorted = df_sorted.reset_index(drop=True)

# Group the dataframe by 'Device' and 'Starting_Frame'
grouped = df_sorted.groupby(grouping_columns)

# Iterate over each group
for _, group in grouped:
    # Extract features and labels
    features = group[feature_columns].values
    labels = group[label_columns].values

    # Create sequences of the specified length
    for start_idx in range(0, len(features) - sequence_length + 1):
        end_idx = start_idx + sequence_length

        # Extract sequence for X and y
        X_seq.append(features[start_idx:end_idx])
        y_seq.append(labels[end_idx - 1])  # Use the label of the last frame in the sequence

    # If the sequence is shorter than the sequence length, pad it
    if len(features) < sequence_length:
        # Pad features with zeros (or another value)
        padded_features = np.pad(features, ((0, sequence_length - len(features)), (0, 0)), 'constant')
        X_seq.append(padded_features)

        # Use the label of the last available frame for the sequence
        y_seq.append(labels[-1])

# Convert to numpy arrays
X_seq = np.array(X_seq)
y_seq = np.array(y_seq)
```

```
# Check shapes
print(f'X_seq shape: {X_seq.shape}')
print(f'y_seq shape: {y_seq.shape}')
```

```
# Train LSTM
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Masking
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.models import load_model

X_train, X_test, y_train, y_test = train_test_split(X_seq, y_seq, test_size=0.1, random_state=42)

# Define model
lstm_model = Sequential()

# Masking layer to handle padded sequences
lstm_model.add(Masking(mask_value=0., input_shape=(X_train.shape[1], X_train.shape[2])))

# LSTM layer with dropout
lstm_model.add(Bidirectional(LSTM(64, return_sequences=False)))
lstm_model.add(Dropout(0.5))

# Output layer
lstm_model.add(Dense(y_train.shape[1], activation='softmax'))

# Compile the model
optimizer = Adam(learning_rate=0.001)
lstm_model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
lstm_history = lstm_model.fit(X_train, y_train, epochs=169, batch_size=16, validation_split=0.2, verbose=0)
lstm_model.save('lstm_model.h5')
lstm_model = load_model('lstm_model.h5')
```

```
# Evaluate the LSTM model on test data
test_loss, test_accuracy = lstm_model.evaluate(X_test, y_test, verbose=0)
print(f'Test Accuracy: {test_accuracy:.2f}')

# Make predictions
y_pred = lstm_model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Classification report
report = classification_report(y_true, y_pred_classes, target_names=label_columns)
print(report)

# Save classification report to a text file
with open('classification_report_lstm.txt', 'w') as f:
    f.write(report)
```

```
# Confusion matrix for LSTM
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import numpy as np

# Compute confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred_classes)

# Plot confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_columns, yticklabels=label_columns)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('LSTM Confusion Matrix')
plt.savefig('confusion_matrix_lstm.png')
plt.show()
```