A Machine Learning Based Long Jump Training Assistant

Jacob Sobota

Abstract

The long jump is a highly technical event in track field where the athlete attempts to jump as far as possible into a sand pit from a running start. Arguably the most important action in this event is the act of jumping at the end of the run, as this is where the athlete's horizontal energy is converted vertically. There are many existing papers that mathematically model the long jump takeoff (the act of jumping) and determine the limb angles, ground angles, and body positions that optimize the jump. There is a general consensus on the features that most considerably affect the quality of the takeoff (and therefore the length of the jump). These features are subtle and nearly impossible to perfectly detect with the human eye. Modern camera technology has allowed many coaches to routinely film the practice jumps of their athletes, but this method still relies on human interpretation. This paper proposes a Long Jump training assistant that uses machine learning to recognize relevant frames of the takeoff, and returns important feature values to be compared to the objective bio-mechanical model.

Introduction

The long jump is a technical event in track and field that requires a combination of speed, strength, and precise technique. While many existing studies have mathematically modeled the long jump takeoff to determine the optimal limb angles, ground angles, and body positions, these features are often subtle and difficult to detect with the human eye. Modern camera technology has made it easier to film the practice jumps of athletes, but analyzing these videos still requires a significant amount of manual effort.

To address this challenge, I propose a Long Jump training assistant that leverages machine learning to recognize relevant frames of the takeoff and return important feature values that can be compared to the objective biomechanical model. This system has the potential to provide coaches and athletes with real-time quantitative feedback on the quality of their jumps, allowing them to make rapid adjustments and optimize their technique for better performance.

Currently, there are no available applications or software that perform these functions in the track and field community. All jump analysis is either performed by the coach on the spot, or videos/data are analyzed with a computer at a later date. Additionally, it is rare for coaches in a training setting to make use of numeric features or existing models, and ones that do are usually operating at a professional level. This in part is due to their access to more advanced tools. This application bridges the gap between those with and without higher technological access, and provides any coach with the means to analyze the long jump takeoff to a superior degree.

The underlying technical goals of this project involve training machine learning models to recognize important positions in the long jump takeoff. This application relies on the performance of two unique models: one that recognizes the frame in which the athlete's takeoff leg first makes contact with the ground during the jump, and another model that recognizes the sequence of frames that make up the takeoff. It is during these two motions that the most important features are extracted.

In this paper, I will present the design and implementation of my Long Jump training assistant, as well as the processes and results of evaluating its effectiveness. Specifically, I will provide a comprehensive explanation of the various stages involved in creating the datasets, building and assessing the machine learning models, and integrating these elements into the final application. I will also discuss the potential impact of this system on long jump training and future improvements and directions for research in this area.

Related Work

An Overview of the Long Jump

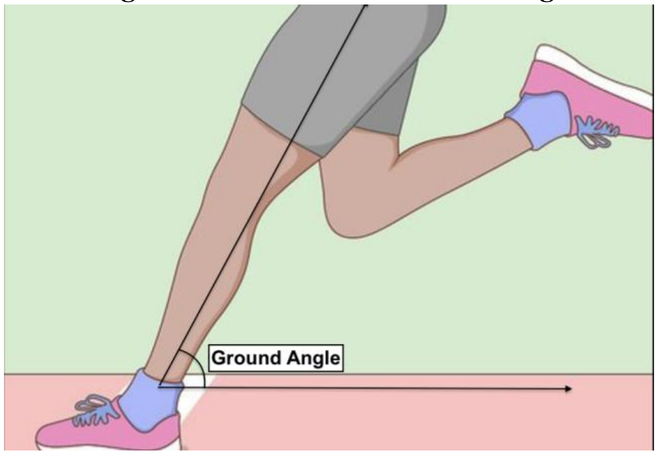
I will first begin with a more detailed explanation of the long jump. This event begins with an approach run. Elite athletes tend to run around 18-22 steps for a full approach. The focus of the approach run is to reach the fastest possible speed where the athlete feels comfortable to jump. This is different than an all-out sprint because pure sprint mechanics are more difficult to jump out of. At the end of the approach run, athletes aim to place their jump foot as close to the foul line as possible without going over, as doing so would rule the jump invalid. The moment the athlete's jump foot makes contact with the ground is known as the touchdown. I will refer to the process of placing down the jump leg, loading it, and jumping off of it as the takeoff. In scientific literature, the takeoff will reference the instant that the foot leaves the ground after jumping. I will use these terms frequently throughout the paper. Once the athlete has taken off, they will employ a variety of techniques to control their bodies in the air. Before landing in the sand, the athlete will extend their arms and legs in front of their body, and drive them back as soon as the feet make contact. The jump distance is measured from the beginning of the foul line to the athlete's closest imprint in the sand.

The takeoff sequence is easily the most technical and the most important part of long jumping. This is where the athlete's horizontal speed is converted into the vertical impulse that lifts them off the ground. The quality of the athlete's takeoff determines the magnitude of the vertical impulse as well as the reduction in horizontal velocity. Ideally, an athlete maximizes the vector sum of the horizontal and vertical component velocities. But what does this look like in the context of the event?

Basic physics shows that the optimum projection angle for an object in free flight is 45 degrees. However, long jumpers' center of masses consistently leave the ground after takeoff at an angle of about 21 degrees, and rarely go higher than 27 degrees (Linthorne). This is due to the fact that a jumper's takeoff speed is not independent of the takeoff angle. The greater the angle of takeoff, the more a jumper must slowdown in order to achieve that angle. As a result, the vector sum of horizontal and vertical velocity is quite small when this angle is closer to 45 degrees. The fact that jumpers tend to keep their takeoff angles around 21 degrees implies that maintaining runway speed through the takeoff is much more important than generating an excessive vertical impulse. This assertion is supported significantly by Lees et al.'s 1994 study, which identifies takeoff speed as the most closely correlated factor with a jumper's distance. This aligns with the findings of numerous other research papers examining the mechanics of the long jump, including Seyfarth, Blickhan, and Van Leeuwen's (2000), Alexander's (1990), and Panoutsakopoulos's (2010).

Touchdown Ground Angle

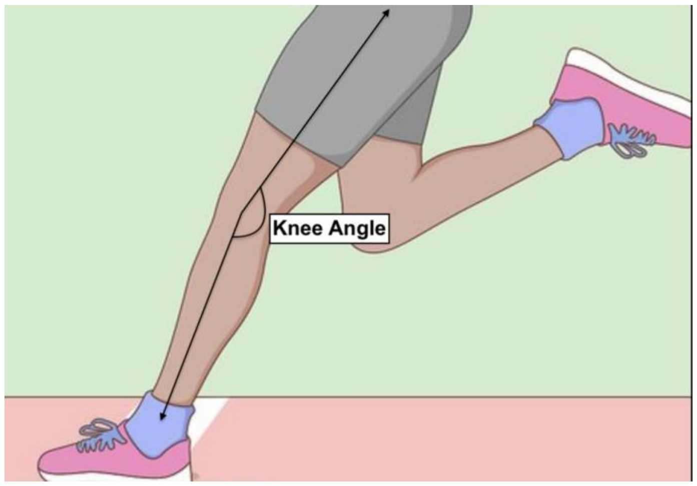
Figure 1: Touchdown Ground Angle



We now know that an athlete's speed when he leaves the ground is the most important factor in jumping far, but how does an athlete carry this speed through the jump? Lees also has an answer for this. In testing for features correlated with takeoff-velocity, he found the feature that offered the largest correlation coefficient was the ground angle at touchdown. The ground angle is the angle that the leg makes with the ground at touchdown. It is measured from the hip, to the ankle, to the horizontal (Figure 1). Thus, athletes who left the ground with the most speed had the most efficient ground angles. With this in mind, many studies have measured the optimal ground angle through mathematical modeling and watching the world's best athletes. There is a general consensus backed observationally and mathematically that the optimal measurement is about 65 degrees, while an acceptable range is anywhere from 60-70 degrees. Conveniently, this measure holds true regardless of the athlete's speed moving into the touchdown. Lisa Bridgett summarizes this phenomenon well in her 2006 paper that analyzes changes in long jump takeoff technique with increasing run-up speed. She finds that over the range of run-up speeds analyzed, the vertical takeoff speed remained constant. In other words, no matter how fast the athlete ran, the vertical impulse remained nearly identical. Bridgett was also in agreement that the optimal ground angle at touchdown is equal to about 65-70 degrees. Therefore, we can assign the 60-70 degree range as the objective "best" ground angle to measure against moving forward.

Touchdown Knee Angle

Figure 2: Touchdown Knee Angle



The athlete's jump-leg knee angle at takeoff is another feature that has an effect on the quality of the jump (Figure 2). The mechanics of this position are simple: a straighter and stiffer leg at touchdown is in a more biomechanically advantageous position to handle load than a bent leg. When the leg is straighter, the moment arm is reduced, meaning less force is required to absorb the force of the jump. A good way to visualize this is to imagine performing a weighted squat. It is much more challenging to hold the weight at the bottom of the squat than at the top. Though it remains a widely discussed feature, the literature does not stress the importance of touchdown knee angle to the extent of touchdown ground angle. The majority of scientists in this field also tend to agree on the same "most efficient" values. Again, Lees et al., 1994 derived from his model of the long jump that the optimal knee angle at touchdown is around 165-170 degrees. Other scientists have introduced more complexity into Lees's model in the years since 1994, and they have all settled on the same range (Panoutsakopoulos 2010, Seyfarth 2000, Graham-Smith 2005, Koyama 2007). In addition to using the model, these studies introduce an observational component. Each of them rely on long jump film from world class athletes (Olympic or World Championship caliber). Therefore, we can conclude both mathematically and observationally that this range is an objective standard at a high level.

Minimum Knee Angle

Many of these papers also measure the change in features over the course of touchdown to takeoff. Examples include measuring angular velocity, change in angle (or angle displacement), and horizontal velocity to name a few. These studies make use of very powerful cameras with high framerate capabilities; cameras that the average coach would likely not use. For that reason, we cannot measure all of these change-over-time features. However, there is still important information we can extract from frame data across the entire takeoff process. The key sequential feature this paper will focus on is the minimum knee angle. Like I discussed above, keeping the knee as straight as possible is important for maintaining a biomechanically advantageous position. However, forces from the ground at touchdown are too powerful for any athlete to maintain their initial touchdown knee angle. The action of reduction in knee angle through the takeoff is consistent in all athletes. However, the minimum angle that the knee reaches is what sets athletes apart. Athletes that tend to be more elastic and stronger eccentrically allow for smaller reductions in their knee angles. Most mathematical models have found that the minimum knee angle should be no less than 135 degrees (Alexander 1990). Moving forward, we will also assume that this minimum is an objective standard for the highest quality jumps.

Overview of Software

There are two key pieces of software that build the foundation of this project: Mediapipe and OpenCV. Mediapipe is an open-source framework developed by Google for building multimodal machine learning applications. It provides a comprehensive set of pre-built solutions for tasks such as hand tracking, pose estimation, object tracking, and facial recognition. This application specifically uses Mediapipe's pose estimation model. This model takes a video as input and predicts the location of the most prominent individual's joint positions. It can then display these joint predictions on top of the video. This is useful for tracking the athlete's body positions through the jump, and then extracting key features such as joint angles and joint distances using OpenCV

OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and machine learning software library. It provides a wide range of tools and functions for image and video processing, including object detection and recognition, feature extraction, tracking, and more. This project specifically utilizes its ability to view videos, interact with videos, and most importantly its compatibility with Mediapipe. OpenCV is what allows the application to display Mediapipe's joint predictions, and it is what made saving the features so simple. Due to its accessible design, the final application model is built in OpenCV as well

Existing Work

Currently, there are no published research papers that use machine learning to recognize relevant long jump frames and sequences. Many papers use Mediapipe or other pose estimation software for a variety of activity classification problems. However, I found no research that applies automatic pose estimation software specifically to track and field. Further, there are no existing papers that leverage pose estimation software to train models or extract features in track and field literature. All previously referenced long jump research papers manually assign joint positions on any video data used, so there is clearly room for improvement in the field. In the semi-relevant papers I found that utilize pose estimation software, I discovered it was uncommon to train activity recognition models to recognize individual frames. The majority of pose estimation models are designed to classify videos based on sequences. However, most videos were the same length and only had a single label throughout the entire video, so they offer no help in identifying frame specific sequences within the video. Thus, the problems this paper attempts to solve are completely novel in this subject area.

Methodology

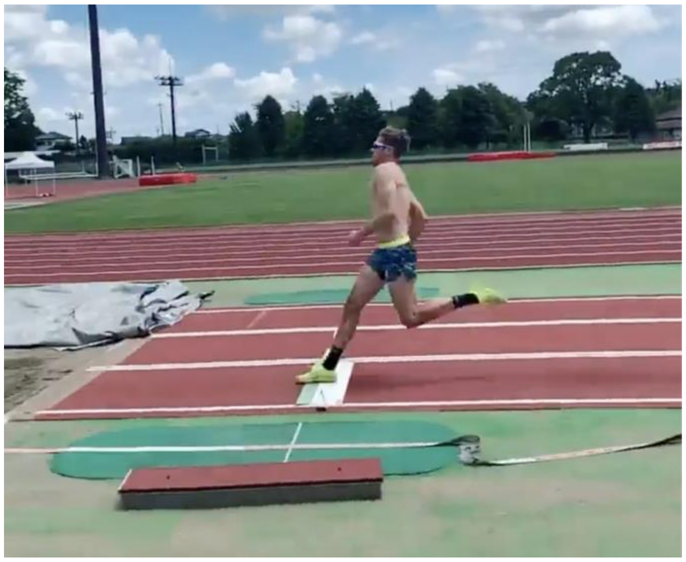
First, I will describe the research design and provide a quick overview of the steps taken. Before beginning the full process of creating the application, I had to find a pose estimation software that could accurately recognize running individuals filmed with a smartphone as that is what coaches tend to use to film their athletes. After testing several publicly available pose estimation softwares, I settled on Mediapipe. Mediapipe offers many features, but I only used the pose estimation software. It is lightweight and posted the most accurate results on the few videos I tested on. Once the pose detection software was selected, I could structure the project. The first step was amassing a collection of Long Jump videos filmed from the proper angle. Next, I built a framework for extracting features from each frame using Mediapipe and OpenCV. After using the feature extractors to build two datasets, I could begin the two machine learning models. One model would be designed to recognize the frame where touchdown first occurs, and one model would be trained to recognize the takeoff as a sequence of frames. Once the models were complete, I could begin combining them into the final application.

Dataset Creation

The base data used in this project comes from 49 different videos of the long jump. In these 49 videos, 23 different long jumpers are represented. Just over half of these videos are training videos taken of the University of Kentucky Track and Field Team courtesy of Kris Grimes, Assistant Track and Field Coach. The rest were downloaded from the Instagram pages of world class long jumpers, and long jump themed accounts (@jumpersworld). There are many long jump videos available online, but I was limited by the following criteria I had set.

In order to best see the long jump takeoff, a person needs to stand perpendicular to the athlete at the takeoff position. In other words, one would need to be in line with the athlete at the instance of the takeoff. A viewpoint closer to where the long jumper starts their run, or in the back of the sand pit, would make it impossible to see the takeoff or to extract any important features. Any experienced coach who films their athletes' jumps will film from a position in line with the takeoff of the jump. A great example of a video that meets all requirements is pictured in Figure 3.

Figure 3: Optimal Film Angle



The next requirement was that the view of the jumper was unobstructed at takeoff. In most training scenarios, there will be no one blocking the jumper at takeoff. There is simply no need for someone to be that close to the runway in front of the camera. In competition however, officials must sit next to the runway at the takeoff board to judge whether the jump was legal or not. As a result, many competition videos are unusable in this context.

I did not place any restrictions on the quality of the video. For an application like this to be generalizable to a large group, it needs to be able to work on both low and high quality videos. However, all videos in the collection are structured so that the athlete is running

left on tape. The vast majority of videos I found were already filmed in this way. In order to make the dataset more complete and to make up for a lack of videos, I horizontally flipped any videos that did not meet this criteria. Ultimately, this does not sacrifice any capabilities of the application as the same features will be extracted regardless of orientation. The only difference is that the perceived takeoff leg will change when the video is flipped.
The final alteration I made for many of the videos is that I trimmed them so they were shorter. This merely cut out some early steps in the run as well as any time spent waiting for the athlete to begin running. This made them more convenient to work with in the early stages of feature extraction.

In addition to the 49 videos used to construct the two models, I downloaded an additional four videos to serve as unseen test examples. These videos also met the same criteria listed above. Unlike many of the 49 videos, these were not trimmed at all and are meant to simulate how an average practice video would look.

Feature Selection

Table 1: Feature values and their descriptions

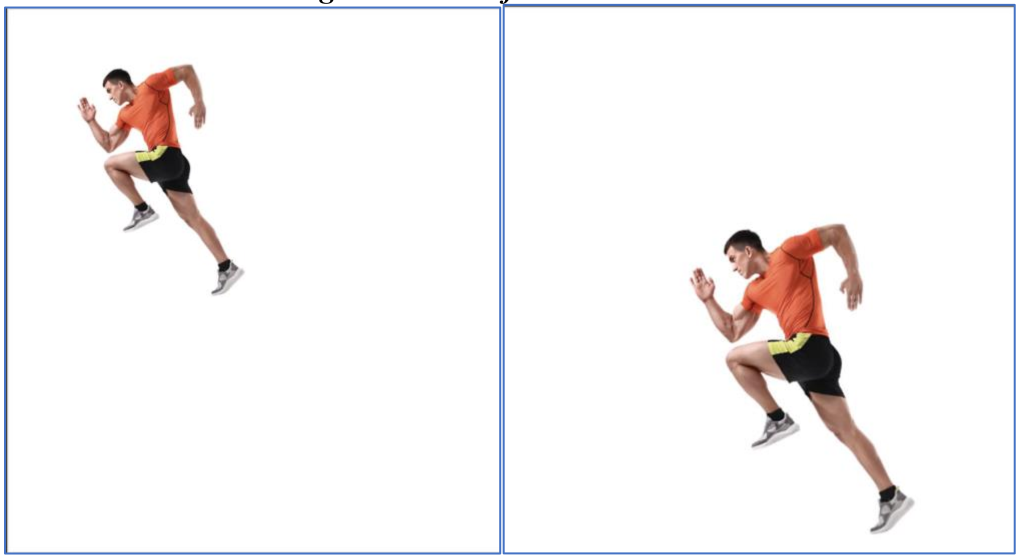
Feature	Description	Justification
Right leg knee angle	Angle formed by right ankle, right knee, right hip	This is an angle of interest that informs the quality of takeoff. See Related Work Section. Used by Lees 1994, Seyfarth 2000, Panoutsakopoulos 2010.
Right hip angle	Angle formed by right knee, right hip, right shoulder	This is an angle of interest that informs the quality of takeoff. Used by Lees 1994, Seyfarth 2000, Panoutsakopoulos 2010.
Left leg knee angle	Angle formed by left ankle, left knee, left hip	This is an angle of interest that informs the quality of takeoff. See Related Work Section. Used by Lees 1994, Seyfarth 2000, Panoutsakopoulos 2010.
Left hip angle	Angle formed by left knee, left hip, left shoulder	This is an angle of interest that informs the quality of takeoff. Used by Lees 1994, Seyfarth 2000, Panoutsakopoulos 2010.
Right foot ground angle	Angle formed by right hip, right ankle, and ground (horizontal)	This is an angle of interest that informs the quality of takeoff. Created by adding an invisible point horizontal to the ankle joint. See Related Work section. Used by Lees 1994, Seyfarth 2000, Panoutsakopoulos 2010.
Left foot ground angle	Angle formed by left hip, left ankle, and ground (horizontal)	This is an angle of interest that informs the quality of takeoff. Created by adding an invisible point horizontal to the ankle joint. See Related Work section. Used by Lees 1994, Seyfarth 2000, Panoutsakopoulos 2010.

Ankle Distance / Leg Length	Distance between each ankle divided by left leg length	Measures distance between each ankle joint. The greater this distance, the more likely the athlete is in a touchdown or takeoff position. Not directly utilized in research, but many papers use ankle distance to the hip, and x-coordinate distance between the two ankles. This combines the two into one measure
X distance between right hip and right ankle / right leg length	The x-coordinate distance from right hip to right ankle divided by right leg length	Measures distance of the ankle in front of the hip. When the value is larger, it indicates that the athlete is reaching the foot far out in front of them, which indicates a takeoff. Measured by Lees 1994
X distance between left hip and left ankle / left leg length	The x-coordinate distance from left hip to left ankle divided by left leg length	Measures distance of the ankle in front of the hip. When the value is larger, it indicates that the athlete is reaching the foot far out in front of them, which indicates a takeoff. Measured by Lees 1994
Right and left hip distance / left leg length	Coordinate distance between right and left hip divided by left leg length	Provides information on the relationship between camera and athlete. When hip coordinates are closer to each other, the athlete is likely in line with camera. If they are farther apart, the athlete is likely facing the camera. This was a personal inclusion that helped model performance
Right hip to right wrist distance / right leg length	Distance from right hip to right wrist divided by right leg length	Athletes often reach positions similar to touchdown while cycling their legs in the air post-takeoff. The main difference between these two positions are arm location. At touchdown, wrists are generally near the hip. In flight, at least one wrist is in the air above the head. Helps the model distinguish between the two. This was a personal inclusion that helped model performance
Left hip to left wrist distance / left leg length	Distance from left hip to left wrist divided by left leg length	Athletes often reach positions similar to touchdown while cycling their legs in the air post-takeoff. The main difference between these two positions are arm location. At touchdown, wrists are generally near the hip. In flight, at least one wrist is in the air above the head. Helps the model distinguish between the two. This was a personal inclusion that helped model performance

Forward foot	Indicates which leg is in front based purely on the x-coordinates	This feature helps the model distinguish left and right foot jumps and their common features. This was a personal inclusion that helped model performance
Takeoff foot	Indicates which leg is on bottom based solely on y-coordinates	This feature is more useful in identifying sequences. Provides information to the sequence model on which leg is the takeoff leg as the takeoff ankle will always be the bottom of the two. This was a personal inclusion that helped model performance

The features chosen for the two machine learning models reflect both the capabilities of Mediapipe and the lack of extensive training data. In practice, the Mediapipe Pose Estimation software represents its joint prediction points as pixel coordinates on top of an image. This is an effective way to view joint predictions in the context of an image, but makes the points impossible to generalize. For example, the following two images contain the exact same figure of a person running (Figure 3). However, the joint coordinates in the first image will be drastically different than the coordinates in the second image due the smaller size of the first figure and its different positioning within the image. A model that receives raw coordinates or even coordinate distance measures will not learn anything because these measurements are specific to the image. In a similar way, jumps that are filmed from up close versus far away would have drastically different coordinate and distance values due to the size and position of the athlete in the final video. This is without even considering the "size" and quality of the videos. In order to solve these potential issues, I chose to measure features that are unaffected by size and positioning within the image. The two primary types of features I use are joint angle measurements, and joint distance ratios. Using features like these, the two images below would return nearly the exact same feature values.

Figure 3: Pair of similar runners



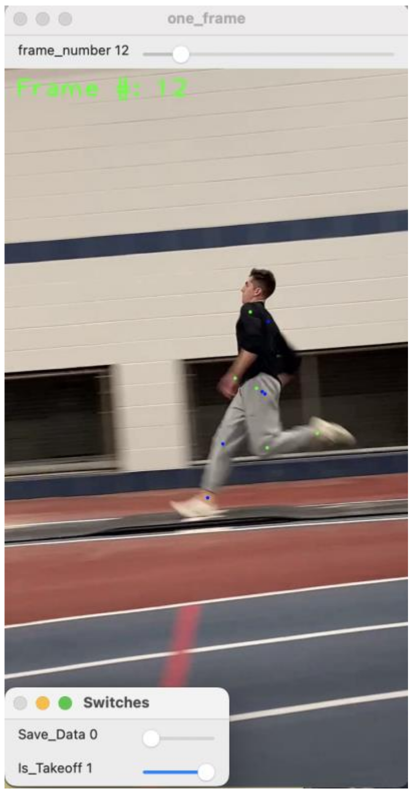
The first features selected were the angles highlighted by the long jump mathematical modelling studies. These are the knee angles, hip angles, and the ground angles. The ground angles are tricky because they are only meaningful when the foot is on the ground, but they are ultimately useful for predicting whether an athlete is in the takeoff position. The next six features are also adapted from the above studies by utilizing relative pixel length. This means that joint point distances are calculated using the raw pixel coordinates, and are then divided by another raw pixel distance. This creates a ratio between two pixel distances rather than the pixel distance itself. The feature I selected in most cases to be the denominator in these ratios was the leg length. The ankle, knee, and hip joints tended to be the easiest for Mediapipe to detect, so leg length remained a reliable measurement. It also provided additional information about the position of the athlete relative to the camera. An athlete's pixel coordinates and point distances change drastically as they move closer to the camera. These changes are recorded in the way that the pixel distance ratios change as the athlete moves through their jump. The distance features I have chosen provide information that generally indicates whether an athlete is in a takeoff position or in a typical running phase. For example, an athlete at touchdown has a very wide split between both ankles. Additionally, the x value displacement between each ankle and its respective hip in touchdown position is much greater than in any phase of the run. For these reasons, the "ankle distance / leg length" and "X distance between ankle and hip / leg length" are appropriate features for indicating a takeoff/touchdown. The final two features are binary variables. The forward foot variable indicates which foot is in front based solely on ankle x coordinate values, and takeoff foot indicates which foot is on the bottom using ankle y coordinate values. Overall, my decision to select these features is based on both existing literature about Long-jump biomechanics, and personal observations/experience in the event.

The second reason I decided on these 14 features was due to the limited amount of video data. With only 49 videos, the model was guaranteed a max of 49 takeoff and touchdown examples. There are obviously many instances throughout each video that are NOT takeoff or touchdown examples, but I wanted to avoid creating a dataset that had an unequal ratio of non-takeoff observations to takeoff observations. Because of this I anticipated the two datasets to be no more than 100 observations. With these reasons in mind, I concluded that the selected 14 features were the most relevant.

In terms of machine learning performance, the model tended to perform better with all of these features included. As discussed in the table, each feature provides a unique piece of information that informs the model as to whether the athlete is in a takeoff/touchdown position or not. Additional research would need to be performed to assess whether more features would benefit this model, or if a larger amount of data would allow for the exclusion of some of these features.

Feature Extraction

Figure 4: Feature Extraction Tool



Feature extraction was performed manually with a framework designed in OpenCV (Figure 4). First, each video was run through Mediapipe's pose estimation software. This software generates a set of joint coordinates for each frame. To create a single object that contains all joint values for every frame, I appended each frame's values into a single list. With this list I created a slider window in OpenCV. This allowed me to cycle through each individual frame with the relevant joint estimation points overlayed (indicated by frame_number in Figure 4). Of the 33 landmarks that Mediapipe predicts, I opted to plot only the coordinates of landmarks that are used in the 14 features. Landmark coordinates that represent the left side of the body are colored lime green, and coordinates representing the right side of the body are colored blue.

Upon selecting a frame using the slider, a hidden Python script computes all the relevant features of that specific frame and assigns them to a list. The tool features two distinct "switches" situated at the lower-left corner of the display. The first toggle, labeled "Save_Data", exports the compiled feature list to a CSV file format on the local computer when the value is set to 1. The second switch allows me to label the class value of the frame. There are two options: the frame contains a takeoff (1), or it does not (0). This value is added as the final variable in the exported list.

With the framework above, building the dataset intended for the touchdown frame recognition model was simple. I saved roughly two sets of feature data from each video: one set of features at touchdown, and one set of features at any other random point in the video. This built out a dataset that had a roughly equal number of touchdown frames and non-touchdown frames. In total, the dataset contains 105 observations and 15 columns.

The process for building the takeoff sequence identifier dataset was more tedious. The framework is almost identical to the previous sequence saver. Due to limitations in OpenCV, I could still only save the features of one frame at a time to the dataset. Since I needed to save these frames in a list together, this was a problem. I decided to add an additional classifier column to solve this issue. Thus, the initial dataset was programmed to hold 3 columns: the frame's feature list, the video number, and the takeoff classification. Since I would only take a max of one of each takeoff classification from each video, the video number and the takeoff classification served as a primary key for each sequence. In other words, all rows that contained the same final two columns were a chronological sequence of frames. With this information, these individual feature lists could be combined later. To fill out this dataset, I went through every video and saved the features for every frame in a takeoff sequence. I also saved a random sequence in each video as reference for the model. The resulting dataset contained 327 rows and 3 columns. Using python, I imported this dataset and concatenated all feature frames into their respective sequences. This created a dataset with 61 rows and 2 columns, where the first column contains a list of lists where each nested list contains the 14 feature values, and the second column contains the takeoff classification label. The reason this dataset has so few examples is due to issues with Mediapipe in accurately identifying full sequences. As I mentioned, the video dataset varies widely in terms of video quality. In many of the lower quality videos, Mediapipe could not accurately identify the relevant joints through the takeoff sequence despite accurately identifying them through most frames. A frequent example of this was that Mediapipe would often confuse the right and left legs throughout the takeoff sequence. In the interest of supplying the model with only accurate information, I opted to exclude takeoff sequences from videos that experienced these glaring errors. With both datasets complete, I was ready to preprocess the data and create the machine learning models.

Model Creation

To begin, I uploaded the touchdown classification dataset into a pandas dataframe in python. The only pre-processing I performed on this dataset was z-score normalization of the angle values. This is due to the fact the angle values were very large compared to the pixel distance ratios. This balanced out the dataset in terms of numerical magnitude and ensured that no feature or feature group would have excessive influence on the model.

Table 2: Machine Learning Model Performance

Model	Test Data Accuracy
Touchdown Recognition Model	80%-91%
Takeoff Sequence Recognition Model	84%-95%

The first model is designed to recognize the frame in which touchdown occurs. Because of the relatively small size of the data, I decided to use a Gaussian Naïve Bayes machine learning model. This model tends to work well with data that has many features but relatively few observations. It is also a probabilistic model. This means that the model returns probability values for each class label rather than the binary class label itself. I used the Guassian Naïve Bayes model from SciKit Learn, an open-source Python library designed for machine learning tasks. To first assess the quality of the model, I trained on the data with a 70/30 train-test split. I repeated this process several times and with multiple random seeds. I consistently achieved a model accuracy percentage in the range of 80-91%. These results are displayed clearly in Table 2. To create the model I would use for the application, I trained on all of the data. I saved this model locally to my computer using joblib, where I could upload and use it at any time.

The second model is designed to recognize the sequence of frames that contain a takeoff in a video of a long jump. The second dataset first required concatenating the features from the individual frames into complete sequences. These sequences were saved as a list of lists, where each object in the list represents a frame. Second, I used the Keras "pad_sequence" function in order to make each sequence of features the same length. I tested the model with multiple max sequence lengths. The longest sequence in the dataset was 14 frames. However, the majority of sequences were around 4 to 7 frames. I trained and tested the model with all of these values. In the end, a max sequence length of 5 gave the best consistent test accuracy value across different random states. The framework used for this data is a Keras Sequential model that utilizes LSTM (Long Short-Term Memory) layers. Sequential models allow for multiple layers in a neural network. Upon completing activation, each layer passes its parameters to the next until the final layer makes a prediction. In this case, each layer consists of an LSTM framework. LSTM's are a type of RNN (recurrent neural network) that possess the ability to selectively forget or retain information from previous inputs, which allows them to handle sequential data particularly well. These networks are commonly used to model video data as videos are sequential by nature. The model utilizes the "adam" (Adaptive Moment Estimation) optimization algorithm for updating weights. This algorithm combines features from stochastic gradient descent and root mean square propagation to become more resistant to noise as well as falling into local minima.

The model itself is made up of three layers. The first layer contains 64 nodes. It takes each sequence of equal length as input. Each individual frame is analyzed and passed on to the second layer after activation. The second layer contains only 16 nodes. This layer processes the sequence as a whole rather than by each frame. This layer returns a single output and passes it on to the third and final layer. This layer is called the "Dense" layer. It's activation is based on calculating a softmax for the input parameter. In other words, it takes the vector output from the previous layer and converts it into a probability distribution over the classes. The class with the highest probability is then assigned by the model.

Like the previous model, I tested with a train/test split of 70/30 and tried a variety of random seeds. This model consistently produced an accuracy rating between 84% and 95%. The model that I created and downloaded for the application was trained on all datapoints, and saved locally using joblib.

Application Build

The application itself is designed to do several things: first, it allows the user to view the video on a slider similar to the feature saver pictured above. Second, it incorporates both machine learning models. Before the slider code is called, the models are run on the video frames and the touchdown frame and takeoff sequence frames are saved. The relevant features are then extracted from these frames. Third, the slider function is called, and the relevant features are displayed conveniently on screen.

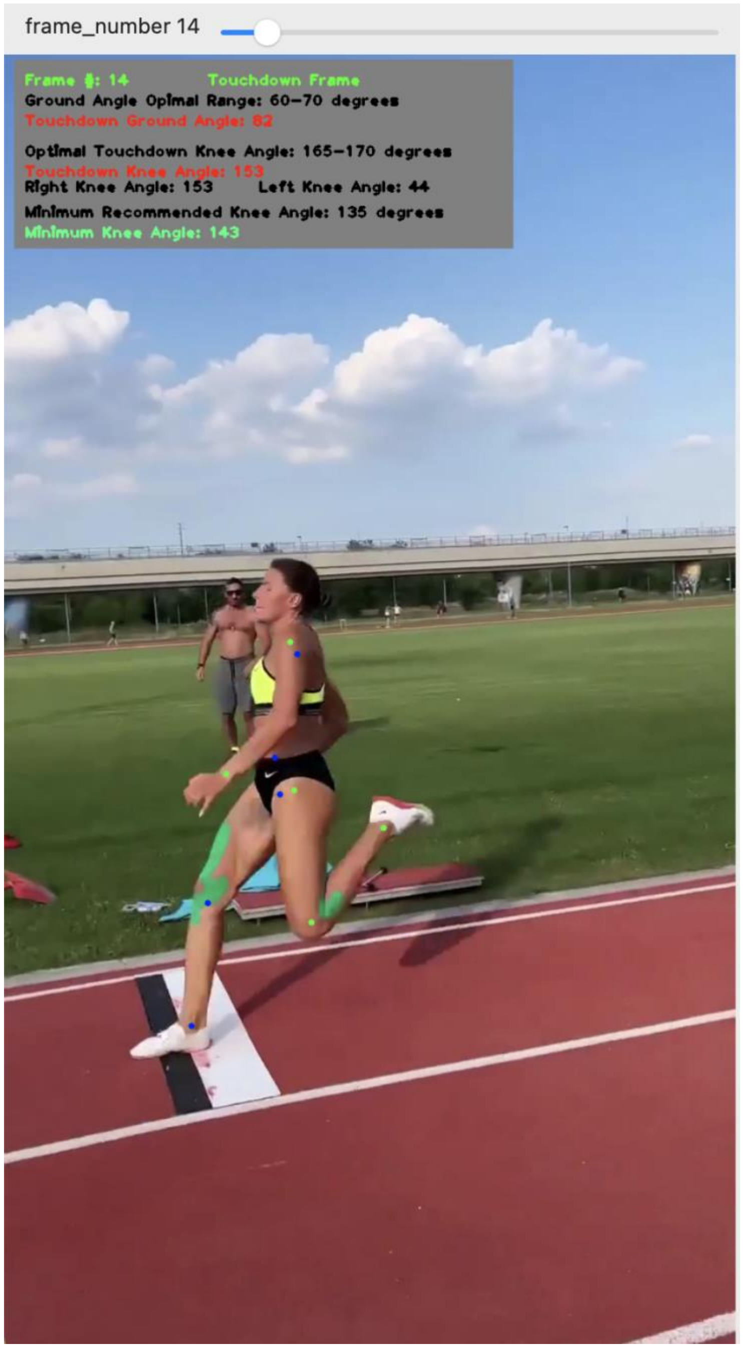
Since the models were not 100% accurate, the process of selecting the proper frames could not be completed using the softmax class assignments. In context of the first model, using softmax causes multiple frames to appear to be touchdown frames. However, by using the predicted probability instead of the softmax binary value, the model can assign the touchdown frame as the one with the highest probability. This eliminates the chance of other similar-looking frames showing up as touchdown frames in the final model. After identifying the touchdown frame, the framework simply has to check which foot is the jump leg, and then extract the relevant features from that frame (knee angle and ground angle). A different process is used for the second model.

The second model takes a sequence of frames as input. Therefore, in order for the model to "recognize" the takeoff sequence, it needs to be fed unseen sequences of equal length. To do this, the model is fed sequences of length 5 frames, which is the max sequence length assigned when the model was trained. The model then skips a frame, and feeds the next 5 frames of the video into the model again. This ensures that the entire video is covered five frames at a time. Since the model is not perfect, there are a few specifications that ensure the proper sequence is returned. It is possible for the sequence identifier to wrongly assign a sequence of frames as a takeoff throughout the video.

To prevent this from disrupting the application, the framework is designed only to return takeoff sequences whose start frame is within 10 frames of the touchdown frame. Since we already know the position of the touchdown frame, we can be certain that our sequence falls very close to the guaranteed touchdown frame, and thus the takeoff itself.

To use the application, a user needs only to enter the file path of the video, run Mediapipe's pose estimation and the model predictions, and finally run the OpenCV slider. These are all contained in one Python script, so it is incredibly easy to use. The end product is a video that the user is able to cycle through to look at different parts of the jump. On the top of the screen, relevant features are listed and color-coded to show whether they are within the "most efficient" ranges discussed earlier in this paper. Like the feature saver, the relevant left side joint estimations are highlighted in green, and the right side joint estimations are blue. A screenshot of the model in use on one of the unseen test videos is shown in Figure 5.

Figure 5: Application Screenshot



```
frame_number 14

Frame #: 14          Touchdown Frame
Ground Angle Opimal Range: 60-70 degrees
Touchdown Ground Angle: 82

Optimal Touchdown Knee Angle: 165-170 degrees
Touchdown Knee Angle: 153
Right Knee Angle: 153      Left Knee Angle: 44
Minimum Recommended Knee Angle: 135 degrees
Minimum Knee Angle: 143
```

All necessary data is included in the textbox at the top of the video. The first line contains the frame number and to the right an indicator that only appears if the user is currently viewing the touchdown frame. Lines 2 and 3 display the optimal ground angle measures and the observed touchdown ground angle measure. If the ground angle is within the recommended range, it appears green, otherwise it appears red. Lines 4,5, and 6 contain the recommended optimal knee angles, the observed touchdown knee angle measurement, and the right and left knee angles of the current frame respectively. The observed touchdown knee angle measurement also appears red when it is outside of the optimal range, and green when it is within the optimal range. The final two lines display the recommended minimum knee angle through the takeoff and the observed minimum knee angle through takeoff. Most of this data remains unchanged as the user adjusts the slider, but the left and right knee angle values update with each frame.

There is no objective way to measure the success of this application other than with the accuracy ratings of the machine learning models. However, we can test the application on the 4 additional unseen long jump videos pulled from Instagram and the University of Kentucky training video cache. Using the application on these videos, the models correctly identified the takeoff frames within one frame on all four videos, and correctly identified enough of each takeoff sequence to extract the minimum knee angle. Ultimately, the first model is a complete success. The second model is effective enough in all test instances, but could stand to be improved in the future.

Discussion

The Long Jump training assistant proposed in this paper has the potential to greatly impact the training and performance of long jump athletes. By leveraging machine learning to recognize relevant frames of the takeoff and return important feature values, coaches and athletes can receive real-time feedback on the quality of their jumps and make rapid adjustments to their technique for better performance. The novelty in this application comes from the fact that the feedback is numeric and based in biomechanical research, whereas coaches tend to exclusively provide qualitative feedback. The introduction of numeric measurements into daily training allow for tracking data over time as well as informing training plans. One example of this could come from analyzing the minimum knee angle. The minimum knee angle is generally a result of the athlete's raw strength, particularly in the quadriceps. Monitoring this value over time could inform strength training plans, as well as track any strength gains in the context of the event.

Despite the overall success of the application, there are certainly some limitations and biases in this study. The largest limitation is the lack of data. In order to obtain a training set of long jump videos for more extensive study, I would likely have to reach out to dozens of coaches to ask them to share training videos from over the years. This would take lots of coordination and time, and would not have been feasible within the confines of a semester. Another side effect of this issue is that all videos are left-facing, and the ones that originally were not were flipped. Creating an application that could recognize videos moving in both directions would have been possible, but also would have added an unknown amount of time to the project.

Additional limitations deal with the quality of the videos in the dataset. Videos with relatively few frames per second are more difficult to classify, especially for the touchdown frame. The model will still recognize the frame that appears to most closely resemble a touchdown frame when the actual instant of touchdown may have gotten lost between frames in the video. The models are trained on some videos with relatively weak framerates. However, this issue is important to note as it may cause slightly incorrect or inconsistent values for the features. The other issue with quality comes from camera angle. This model will not work unless the videos are filmed as parallel with the takeoff as possible. The application may still recognize the correct frames when filming from another position, but the feature values returned will not be as accurate.

If I were to continue working on this project, the first piece of additional research I would introduce would be applying an accurate 3-dimensional pose estimation software. This means that from a 2-dimensional video, the software would produce accurate 3-dimensional joint estimates. Software like this would eliminate the need to be parallel with the takeoff. The coach could film from any angle and the same joint estimates would be returned. There is currently a version of Mediapipe pose estimation that returns 3-dimensional coordinates, but I found it to be too inaccurate for the purposes of this project. I would ideally find and test the highest quality 3D pose estimation software. Once I had the best one, I would look to recreate this application. Since more camera angles would be available, there would be more data to train and test. In addition, the application would be more generalizable.

Conclusion

In conclusion, the long jump training assistant proposed in this paper has the potential to revolutionize the way long jump athletes train and improve their performance. By using machine learning to recognize important frames and extract feature values in the takeoff sequence, this application provides coaches and athletes with real-time feedback on the quantitative quality of their jumps.

Furthermore, this application bridges the gap between those with and without advanced technological access, allowing any coach to analyze the long jump takeoff to a superior degree. This refinement of this application over time has the potential to democratize access to advanced training methods, improving the overall competitiveness of the sport.

Future directions for research in this area could involve incorporating additional machine learning models to recognize other important features in the long jump, as well as expanding this technology to other events in track and field. With the advancements in camera technology and machine learning, there is immense potential for these tools to improve athlete performance and training across a wide range of both track events and sports in general.

References

1. Lees, A., Fowler, N., & Derby, D. (1993). A biomechanical analysis of the last stride, touch-down and take-off characteristics of the women's long jump. *Journal of sports sciences*, *11*(4), 303–314. https://doi.org/10.1080/02640419308730000
2. Alexander, R.M. (1990). Optimum take-off techniques for high and long jumps. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences, 329 1252*, 3-10 .
3. Bridgett, Lisa & Linthorne, Nicholas. (2006). Changes in long jump take-off technique with increasing run-up speed. Journal of sports sciences. 24. 889-97. 10.1080/02640410500298040.
4. Panoutsakopoulos, Vassilios & Papaiakovou, Georgios & Katsikas, Fotios & Kollias, Iraklis. (2010). 3D Biomechanical Analysis of the Preparation of the Long Jump Take-Off. New studies in athletics. 25. 55-68.
5. Linthorne, Nick. " Optimum Take-Off Angle in the Long Jump." *Long Jump Optimum Take-off Angle,* https://www.brunel.ac.uk/~spstnpl/BiomechanicsAthletics/LJOptimumAngle.htm#:~:text=The%20optimum%20take%2Doff%20angle%20for%20the%20athlete%20is%20obtained,%C2%B0%2C%20but%20about%2022%C2%B0.
6. Google. (2021). Mediapipe: A framework for building multimodal (cross, face, hand, iris) applied machine learning pipelines. https://google.github.io/mediapipe/solutions/pose.html
7. OpenCV. (2021). OpenCV: Open Source Computer Vision Library. https://opencv.org/
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830. https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf
9. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16) (pp. 265-283). USENIX Association. https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
10. Chollet, F., & Keras Team. (n.d.). Video classification with pre-trained models. Keras. Retrieved April 13, 2023, from https://keras.io/examples/vision/video_classification/
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Naive Bayes. scikit-learn. Retrieved April 13, 2023, from https://scikit-learn.org/stable/modules/naive_bayes.html