Using Machine Learning to Model the Human Run

[redacted]

[redacted]

AP Capstone: Research

May 1, 2023

Word Count: 4,918

Abstract

This research project addresses the challenge of analyzing the form of a human run without the use of sensors attached to the athlete's body by proposing a new analysis method. The method involves using computer vision to collect data about an athlete's run and analyzing it using a machine learning model that is based on a professional runner. The goal of the machine learning model is to correct the form of an athlete in an inputted video by making predictions about their motion. Through analyzing the motion of an Olympic runner with the Spearman Rank correlation test, the research concludes that modeling the run with machine learning is feasible. The creation of said machine learning algorithm is judged successful with limited accuracy. The potential application of the model in the real world is judged based on Paul Fleming's findings in "Athlete and coach perceptions of technology needs for evaluating running performance" (2018).

Literature Review

The challenge of maximizing training efficiency has been tackled by running coaches for as long as running has formally existed. As technology continues to advance, it is drastically reshaping the field of training, with a multitude of new devices and tools being used to take on the challenge. Wearable technologies and virtual reality are among the most popular developing technologies for self training and improvement. For runners, a lack of the right tool could be the only hindrance holding improvement back. If the right technologies are provided to greater numbers of people, however, aspiring runners of all skill levels and affluence could have the opportunity to improve their performance. These technologies are already commonly used by athletes in a variety of other sports and are expanding to new athletic endeavors like running rapidly. The goal of this project is to propose a machine learning-based training tool with the ability to give feedback to a runner regarding their form, more specifically the exact positioning of their body during a run, and to evaluate its effectiveness at providing data that is useful in the coaching process.

In recent years, the dominating technologies have been wearables, defined as “digital products that can take the form of accessories or clothing” (Wei et al., 2021). Borowski-Beszta and Polasik discuss the practical use cases of wearable devices in “Wearable Devices: New Quality in Sports and Finance”, in which they claim, “The practical applications of wearables are dominant. The devices are popular primarily among users who practice sports and want to monitor their progress, body parameters or their life balance” (Borowski-Beszta & Polasik, 2020). Smartwatches see the most use out of any devices in the wearable technology category. They have asserted their dominance in the field due to their convenience and ease of use. One

can easily slip on a smartwatch without second thought and wear it throughout the day, allowing it to track vital information about an athlete, including their heart rate, steps, and even blood oxygen level. Although practical, smartwatches don't have the ability to detect the exact position of the athlete wearing them, limiting their ability to assess specific running form.

Wearables can also take the form of full-body systems that are worn to detect the exact position of an athlete in 3d space. In reference to one such example of a full body system aimed at tracking a golfer's swing, MySwing Professional, Shiqing Wei explains that, "The precise positioning system can capture the player's movements accurately since the player wears multiple micromotion sensors" (Wei et al., 2021). Lying at the other extreme, full body motion capture systems are costly and unavailable to most athletes, leading to a need for a middle ground.

Virtual reality (VR) is a training method that hasn't seen nearly as much widespread adoption as smartwatches and full body sensors. Its appeal, however, comes not just from the accuracy and specificity of data that it is able to collect, but also its ability to analyze this data and report it to a user in real time. Similar to full body motion tracking systems, virtual reality has increased accuracy over the naked eye (Lv et al., 2022), although with the need for a bulkier system. Virtual reality is a method used by researchers as a means to collect data about an athlete as well as provide direct feedback in a headset that is worn by a user, the key difference being the ability to issue this feedback instantly. When using virtual reality, the athlete's body position in space, as well as trajectory, are tracked (Lv et al., 2022). This deeper analysis of movement from VR allows athletes to gain a fuller understanding of their strengths and weaknesses and improve at a faster pace. The authors of, "Application of Virtual Reality Technology Based on Artificial

Intelligence in Sports Skill Training", conclude, "after using the artificial intelligence virtual reality technology simulation system, the effect of athletes in sports training is better; the action standard is longer; the mastery degree is better; the time to reach the same level is shorter, that is, the action efficiency is higher; and the application of the system can effectively improve the level of athletes' sports training skills" (Lv et al., 2022). However, the use of VR is not practical because the equipment involved in a full VR system is extensive, including sensors and a headset that are heavy and clunky, inhibiting the movement of any athlete that may use the technology. This means that although the data provided by VR systems is far more accurate in reporting on an athlete's form than a smartwatch, it does not give an accurate representation of an athlete's performance.

Reviewing the most common technologies used by other researchers in this field establishes the gap of creating a running training tool that combines an input method that doesn't inhibit athlete movement and a novel analysis method. Discussed below, my method utilized computer vision for input and deep learning for analysis.

In order to collect data about a subject's position in space without the need for a cumbersome headset or other wearable sensors that inhibit movement, the proposed method involves utilizing computer vision as a data collection medium. By moving the data collection medium away from the body of the runner and to an external camera, the athlete is no longer restrained by a heavy headset or restrictive full body sensors. A large amount of data can still be collected from this external vantage point, from which a camera can produce a two-dimensional image. This limits the pose reconstruction to two dimensions, as three-dimensional reconstruction is not yet accurate enough for this use case (Yiannakides et al., 2019).

Furthermore, using computer vision as the input method, thus reducing the amount of necessary equipment, would improve a running coach's ability to train athletes and allow them to train more of them due to improved affordability and accessibility. For the same reason, the tool could also allow athletes to identify problem areas in their running on their own. The Tensorflow-powered BlazePose model was used to collect data about the human pose. This model, similarly to the commonly used PoseNet model has the ability to, "realize the pose estimation of human body movements, facial expressions, finger movements and so on" (Zhou et al., 2021). This model was chosen because it is built using Tensorflow, a machine learning framework created by Google that has a number of features that this project made use of, including pre built models such as BlazePose, the ability to create and train models with relative ease due to being widely used and well-documented, and a web implementation that allows the creation of user interfaces. Machine learning is the field in computer science that involves allowing computers to find patterns in data. Predictions can then be made about any new, unseen data by using machine learning models. BlazePose, for example, is a model that takes image data as input and outputs a set of coordinate pairs, each corresponding to one key point on the subject's body. It is able to do this because it has "seen" many images of people with their pose annotated by humans. By learning to understand patterns in the images, it gained the ability to predict the pose of subjects in new images without human assistance.

The data generated with the pose estimation model was analyzed by a newly created neural network, or NN, that was made using Tensorflow. Neural networks are machine learning models inspired by biological brain function, allowing them to "learn" (Cser et al., 2001). Neural networks excel at creating associations between data that humans aren't realistically able to

create. This would allow a neural network to find connections between a runner's pose, time, limb length, body shape, and other data points that are able to be collected with computer vision. Josef Cser discusses the use of multiple neural networks in conjunction for increased versatility (Cser et al., 2001). This method of combination was used in my method in order to utilize separate networks for input (pose estimation) and analysis (newly created NN). Once fitted to a dataset, a NN is able to predict an output given certain input. A dataset of poses over the course of a professional runner's run can be used to fit the NN, allowing it to predict a runner's position at a certain point in time. This correct position can be compared to a runner's observed position with a function such as cosine similarity in the sklearn library (Zhou et al., 2021). By applying the NN that is trained on a professional runner to any non-professional runner, its predictions essentially become corrections, allowing the model to show how a runner's form should be changed and improved.

Finally, the data should be able to be presented to the user in a digestible format. Because Tensorflow was used to create the model, an interface can be created on the web with Tensorflow.js (Smilkov et al., 2019), using the difference between expected and observed pose to identify how running form should be improved and communicate it to a user. Data was collected from an Olympic runner and a neural network was created to model their run in order to answer the research question: To what extent can machine learning be used to model the human run for use as a tool in athletes' training?

Methodology

In order to explore the feasibility of adapting a machine learning (ML) approach to running improvement, a correlation should exist between the position of a runner's various key

points throughout the run, allowing the run to be modeled and predicted with a neural network. Because NNs make predictions by finding patterns, having strong correlations among the motion of the key points provides strong patterns for the NN to find, allowing it better understand what the proper run looks like (Krogh, 2008). The ML model then must be created to explore and evaluate the application of the technology. When deciding on a ML framework, it was necessary to consider a few important criteria: The framework must be well documented for ease of use, it must have a web implementation for the creation of a user interface, and it must be freely available. Google's Tensorflow is a ML framework that satisfies all of these criteria. Furthermore, Tensorflow includes BlazePose, a prebuilt deep learning (a subset of ML) model created for human pose detection, a crucial piece for analyzing a human run. BlazePose is used to attain the coordinates of a runner's key joints on a two dimensional plane when viewed perpendicular to the forward motion of the runner. The evaluation research process is appropriate in analyzing the feasibility and effectiveness of a newly proposed product. As such, the approach is non-experimental. Because the data would be analyzed numerically, the research used a quantitative method.

Procedure

To analyze the correlation between joints during a "perfect" run, a video clip was chosen based on the following criteria: The runner should be top performing, the video must be recorded from an angle perpendicular to the trajectory of the runner, there must be ample contrast between the color of the runner's clothing and the color of the background, increasing the accuracy of BlazePose estimations, and the clip must include two full strides of a run. It wasn't necessary for the footage to be recorded at a certain distance from the subject, as the coordinates would be

normalized. Maintaining a consistent perspective was a crucial component for creating a more accurate model faster. Although it is possible for a model to learn to differentiate between different perspectives, providing feedback accordingly, it wasn't feasible within the time constraints of the research project. Furthermore, looking at only one perspective eliminates the third dimension, simplifying all data onto a 2d plane. Ultimately, the clip was selected from the 5000m final at the 2016 Summer Olympics, in which [redacted] is shown running under all of the given criteria. Containing 33 frames, the clip adequately included more than the two necessary full strides. BlazePose was then implemented for all 33 frames utilizing the Tensorflow JS library, saving the output of each frame to a file for analysis using Spearman Rank Correlation as well as for use as a dataset in the creation of the ML model. For a complete pose, BlazePose generates an array of 33 keypoints and their coordinates. Each keypoint array was appended to another array, storing all of the data for each frame in order. The coordinates of each point were left in pixel units in this dataset, but they would be normalized in the ML model itself. See Appendix A for the relevant Javascript implementation of these steps.

For a ML model to be created, or more specifically a deep learning model, it must be fit to relevant data. The dataset created previously was used for this purpose. Fitting is the process of allowing a model to progressively modify its weights, or the parameters that allow it to predict an output, until it produces an expected result based on a given input.

Next, the deep learning model itself needed to be created. For this, I used Tensorflow and Keras with the Python programming language. Keras allows models to be built very quickly with the inclusion of prebuilt layers that are simple to implement. In a deep learning model, data flows through layers of neurons, also called parameters or weights. As data flows through the network,

each layer of neurons modifies the input data in some way, with each layer type having a different effect on the data. The model created in this research involved two types of layers, a normalization layer and the most commonly used dense layer. The normalization layer converts the inputted data from pixel units to values between 0 and 1. Doing this does not modify the meaning of the data. Instead, it simply formats it in a way that is easier for the NN to work with. Dense layers are typically the layers with most of the weights that are modified during fitting. They can be thought of as the layers that "learn". For the purpose of this research project, five dense layers were used with a size of 512 weights each. Finally, the model needed to be fit to the dataset. To do so, the data was split into a training dataset and a test dataset, with 60% of the data points being used for training and 40% being used for testing. Each dataset was split further into a set of features and labels. The features are the coordinates of the points that the model would predict, while the labels are the desired predictions, i.e. the coordinates that describe the pose one frame into the future. The training dataset was used to fit the model, using the mean absolute error loss function, a widely used and commonly agreed upon loss function. The goal of the neural network is to make changes to its parameters that lower the loss function, creating the feedback loop necessary for the network to "learn" by taking incremental steps in the right direction (Song & Urtasun, 2016). The final code for the model can be found in Appendix B.

The fitted model was then able to predict the locations of all of a runner's joints as many frames into the future as desired based on an inputted pose. For the model to be usable and demonstrable, it needed to be implemented into a user interface, or UI. This was created with standard web development tools (HTML, CSS, Javascript) and Tensorflow.js. Its functionality didn't need to be complex, as the only requirement was to display the poses of newly predicted

frames. The UI should simply take a video as input, process the video with a combination of BlazePose and the newly built deep learning model, and display a summary of the least and most accurate points in the run. Processing the video was done very similarly to preparing the dataset. Coordinates were detected with BlazePose for each frame, after which the custom model was used to predict the position of the runner 15 frames into the future (about 1 stride or 0.5 seconds), predicting in 15-frame intervals until the entire video clip has been predicted. Then, the predicted position at each frame was compared to the actual position using the cosine similarity function from the Keras library, as used by Zhou et al. to calculate the similarity between two human poses. This function returns a value from -1 to 1 that tells the similarity between two data points, in this case checking the similarity between the predicted and actual positions. Since the predictions were based on ideal running form (from a professional runner), the similarity suggests how good the inputted form is in comparison. This value was calculated at every frame, allowing the UI to display the parts of the video which had the lowest accuracy of running form, which are the frames with a cosine similarity closest to 1. To do so, the user is able to scrub through a video player that shows the accuracy of their run on that frame as a percentage (as calculated using cosine similarity), and a skeleton overlay is displayed to show the expected pose at that point.

Results and Discussion

The data collected reveals the x and y components of the position of thirteen key points on the human body over the course of a stride. The key points used are as follows: Nose (representative of head position), shoulders, elbows, wrists, hips, knees, and ankles. The results

of collecting the Spearman Rank correlation between each combination of the positions of these points is presented in Figure 1.

Key Points	nose_x	nose_y	left_shoulder_x	left_shoulder_y	right_shoulder_x	right_shoulder_y	left_elbow_x	left_elbow_y	right_elbow_x	right_elbow_y	left_wrist_x	left_wrist_y	right_wrist_x	right_wrist_y	left_hip_x	left_hip_y	right_hip_x	right_hip_y	left_knee_x	left_knee_y	right_knee_x	right_knee_y	left_ankle_x	left_ankle_y	right_ankle_x	right_ankle_y
nose_x	1.00	-0.51	0.68	-0.54	0.92	-0.46	0.38	-0.38	0.63	-0.18	0.62	-0.22	0.66	-0.12	0.90	-0.10	0.91	-0.05	0.37	0.23	0.20	0.18	0.31	0.33	0.05	-0.20
nose_y	-0.51	1.00	-0.34	0.95	-0.51	0.95	-0.19	0.69	-0.32	0.67	-0.33	0.55	-0.36	0.66	-0.57	0.75	-0.55	0.71	-0.26	0.49	-0.02	0.53	-0.22	-0.23	0.15	0.32
left_shoulder_x	0.68	-0.34	1.00	-0.40	0.40	-0.34	0.76	-0.34	0.08	-0.28	0.68	-0.12	0.16	0.06	0.75	-0.03	0.78	-0.16	-0.22	0.27	0.70	-0.12	-0.34	-0.18	0.63	0.29
left_shoulder_y	-0.54	0.95	-0.40	1.00	-0.50	0.94	-0.23	0.75	-0.33	0.67	-0.29	0.55	-0.34	0.64	-0.62	0.79	-0.61	0.73	-0.24	0.54	-0.05	0.53	-0.15	-0.14	0.05	0.25
right_shoulder_x	0.92	-0.51	0.40	-0.50	1.00	-0.44	0.14	-0.33	0.78	-0.09	0.47	-0.25	0.78	-0.19	0.77	-0.12	0.74	-0.01	0.59	0.18	-0.13	0.26	0.61	0.55	-0.29	-0.44
right_shoulder_y	-0.46	0.95	-0.34	0.94	-0.44	1.00	-0.22	0.78	-0.21	0.82	-0.33	0.64	-0.22	0.66	-0.47	0.82	-0.52	0.80	-0.12	0.56	-0.07	0.66	-0.11	-0.27	0.12	0.37
left_elbow_x	0.38	-0.19	0.76	-0.23	0.14	-0.22	1.00	-0.33	-0.11	-0.19	0.83	-0.44	-0.02	-0.05	0.53	-0.05	0.49	-0.21	-0.35	0.06	0.65	-0.23	-0.50	-0.28	0.57	0.40
left_elbow_y	-0.38	0.69	-0.34	0.75	-0.33	0.78	-0.33	1.00	-0.04	0.73	-0.38	0.63	-0.03	0.42	-0.41	0.77	-0.50	0.75	0.06	0.48	-0.16	0.63	0.04	-0.20	-0.04	0.34
right_elbow_x	0.63	-0.32	0.08	-0.33	0.78	-0.21	-0.11	-0.04	1.00	0.30	0.21	-0.18	0.97	-0.44	0.61	-0.04	0.39	0.16	0.92	-0.01	-0.54	0.53	0.81	0.30	-0.54	-0.22
right_elbow_y	-0.18	0.67	-0.28	0.67	-0.09	0.82	-0.19	0.73	0.30	1.00	-0.17	0.49	0.31	0.30	-0.11	0.71	-0.32	0.79	0.34	0.41	-0.33	0.87	0.27	-0.29	-0.11	0.39
left_wrist_x	0.62	-0.33	0.68	-0.29	0.47	-0.33	0.83	-0.38	0.21	-0.17	1.00	-0.53	0.31	-0.18	0.67	-0.09	0.59	-0.17	-0.03	0.08	0.41	-0.08	-0.13	0.03	0.25	0.14
left_wrist_y	-0.22	0.55	-0.12	0.55	-0.25	0.64	-0.44	0.63	-0.18	0.49	-0.53	1.00	-0.17	0.69	-0.28	0.58	-0.20	0.58	-0.14	0.59	0.09	0.50	-0.06	-0.17	0.24	0.18
right_wrist_x	0.66	-0.36	0.16	-0.34	0.78	-0.22	-0.02	-0.03	0.97	0.31	0.31	-0.17	1.00	-0.44	0.66	-0.03	0.44	0.14	0.87	0.02	-0.45	0.53	0.75	0.23	-0.45	-0.13
right_wrist_y	-0.12	0.66	0.06	0.64	-0.19	0.66	-0.05	0.42	-0.44	0.30	-0.18	0.69	-0.44	1.00	-0.28	0.62	-0.08	0.52	-0.48	0.76	0.39	0.33	-0.32	0.00	0.42	0.05
left_hip_x	0.90	-0.57	0.75	-0.62	0.77	-0.47	0.53	-0.41	0.61	-0.11	0.67	-0.28	0.66	-0.28	1.00	-0.18	0.90	-0.13	0.37	0.06	0.25	0.12	0.17	0.01	0.20	0.09
left_hip_y	-0.10	0.75	-0.03	0.79	-0.12	0.82	-0.05	0.77	-0.04	0.71	-0.09	0.58	-0.03	0.62	-0.18	1.00	-0.20	0.93	-0.06	0.80	0.13	0.71	-0.08	-0.13	0.18	0.34
right_hip_x	0.91	-0.55	0.78	-0.61	0.74	-0.52	0.49	-0.50	0.39	-0.32	0.59	-0.20	0.44	-0.08	0.90	-0.20	1.00	-0.20	0.11	0.15	0.46	-0.03	0.00	0.13	0.34	-0.05
right_hip_y	-0.05	0.71	-0.16	0.73	-0.01	0.80	-0.21	0.75	0.16	0.79	-0.17	0.58	0.14	0.52	-0.13	0.93	-0.20	1.00	0.18	0.68	-0.09	0.82	0.12	-0.05	-0.01	0.25
left_knee_x	0.37	-0.26	-0.22	-0.24	0.59	-0.12	-0.35	0.06	0.92	0.34	-0.03	-0.14	0.87	-0.48	0.37	-0.06	0.11	0.18	1.00	-0.14	-0.75	0.51	0.88	0.29	-0.71	-0.26
left_knee_y	0.23	0.49	0.27	0.54	0.18	0.56	0.06	0.48	-0.01	0.41	0.08	0.59	0.02	0.76	0.06	0.80	0.15	0.68	-0.14	1.00	0.33	0.58	-0.04	0.09	0.27	0.09
right_knee_x	0.20	-0.02	0.70	-0.05	-0.13	-0.07	0.65	-0.16	-0.54	-0.33	0.41	0.09	-0.45	0.39	0.25	0.13	0.46	-0.09	-0.75	0.33	1.00	-0.33	-0.82	-0.31	0.88	0.39
right_knee_y	0.18	0.53	-0.12	0.53	0.26	0.66	-0.23	0.63	0.53	0.87	-0.08	0.50	0.53	0.33	0.12	0.71	-0.03	0.82	0.51	0.58	-0.33	1.00	0.45	0.01	-0.21	0.14
left_ankle_x	0.31	-0.22	-0.34	-0.15	0.61	-0.11	-0.50	0.04	0.81	0.27	-0.13	-0.06	0.75	-0.32	0.17	-0.08	0.00	0.12	0.88	-0.04	-0.82	0.45	1.00	0.56	-0.85	-0.57
left_ankle_y	0.33	-0.23	-0.18	-0.14	0.55	-0.27	-0.28	-0.20	0.30	-0.29	0.03	-0.17	0.23	0.00	0.01	-0.13	0.13	-0.05	0.29	0.09	-0.31	0.01	0.56	1.00	-0.62	-0.93
right_ankle_x	0.05	0.15	0.63	0.05	-0.29	0.12	0.57	-0.04	-0.54	-0.11	0.25	0.24	-0.45	0.42	0.20	0.18	0.34	-0.01	-0.71	0.27	0.88	-0.21	-0.85	-0.62	1.00	0.65
right_ankle_y	-0.20	0.32	0.29	0.25	-0.44	0.37	0.40	0.34	-0.22	0.39	0.14	0.18	-0.13	0.05	0.09	0.34	-0.05	0.25	-0.26	0.09	0.39	0.14	-0.57	-0.93	0.65	1.00

Figure 1: Spearman rank correlation between all key points.

Spearman's rank correlation was used to find the nonlinear correlation between every combination of variables. The resulting correlation coefficients were used to interpret the strength of the correlation between each pair of position variables. A value of 1 represents a perfect correlation, a value of -1 represents a perfect negative correlation, and a value of 0 represents no correlation, or the null hypothesis in this case. Highlighted in green are the values with an absolute value greater than 0.6, representing a moderate correlation strength. Because this research deals with the motion of an imperfect biological organism, a human, 0.6 can be accepted as the threshold for correlation. The subsequent limitations of this fact will be discussed, as the strength of the correlation has a direct impact on the accuracy of any attempt to

model the run. The p-values from the Spearman rank correlation tests (shown in Figure 2) were used to validate the statistical significance of the found correlations. The widely agreed upon α-value of 0.05 was used in this validation. Any p-value above α represents a coordinate pair without statistically significant correlation. In other words, pairs with p-values above 0.05 are not strongly related. Therefore, all p-values below this α-value represent pairs of coordinates with a statistically significant correlation whose strength is determined by the corresponding correlation coefficient. These are highlighted in green in Figure 2.

Key Points	nose_x	nose_y	left_shoulder_x	left_shoulder_y	right_shoulder_x	right_shoulder_y	left_elbow_x	left_elbow_y	right_elbow_x	right_elbow_y	left_wrist_x	left_wrist_y	right_wrist_x	right_wrist_y	left_hip_x	left_hip_y	right_hip_x	right_hip_y	left_knee_x	left_knee_y	right_knee_x	right_knee_y	left_ankle_x	left_ankle_y	right_ankle_x	right_ankle_y
nose_x	0.00	0.00	0.00	0.00	0.00	0.01	0.03	0.03	0.00	0.33	0.00	0.23	0.00	0.52	0.00	0.60	0.00	0.78	0.03	0.21	0.28	0.33	0.09	0.06	0.78	0.28
nose_y	0.00	0.00	0.05	0.00	0.00	0.00	0.30	0.00	0.07	0.00	0.06	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.91	0.00	0.22	0.21	0.41	0.07
left_shoulder_x	0.00	0.05	0.00	0.02	0.02	0.06	0.00	0.06	0.68	0.13	0.00	0.52	0.37	0.73	0.00	0.88	0.00	0.40	0.23	0.13	0.00	0.52	0.05	0.32	0.00	0.11
left_shoulder_y	0.00	0.00	0.02	0.00	0.00	0.00	0.21	0.00	0.07	0.00	0.11	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.19	0.00	0.77	0.00	0.41	0.43	0.79	0.16
right_shoulder_x	0.00	0.00	0.02	0.00	0.00	0.01	0.44	0.07	0.00	0.63	0.01	0.17	0.00	0.29	0.00	0.50	0.00	0.97	0.00	0.32	0.48	0.14	0.00	0.00	0.10	0.01
right_shoulder_y	0.01	0.00	0.06	0.00	0.01	0.00	0.24	0.00	0.25	0.00	0.07	0.00	0.22	0.00	0.01	0.00	0.00	0.00	0.50	0.00	0.71	0.00	0.56	0.13	0.52	0.04
left_elbow_x	0.03	0.30	0.00	0.21	0.44	0.24	0.00	0.07	0.54	0.30	0.00	0.01	0.93	0.78	0.00	0.80	0.00	0.25	0.05	0.75	0.00	0.20	0.00	0.12	0.00	0.02
left_elbow_y	0.03	0.00	0.06	0.00	0.07	0.00	0.07	0.00	0.81	0.00	0.03	0.00	0.87	0.02	0.02	0.00	0.00	0.00	0.76	0.01	0.39	0.00	0.83	0.27	0.82	0.06
right_elbow_x	0.00	0.07	0.68	0.07	0.00	0.25	0.54	0.81	0.00	0.10	0.25	0.32	0.00	0.01	0.00	0.83	0.03	0.38	0.00	0.94	0.00	0.00	0.00	0.09	0.00	0.22
right_elbow_y	0.33	0.00	0.13	0.00	0.63	0.00	0.30	0.00	0.10	0.00	0.34	0.00	0.08	0.10	0.53	0.00	0.07	0.00	0.06	0.02	0.06	0.00	0.14	0.11	0.55	0.03
left_wrist_x	0.00	0.06	0.00	0.11	0.01	0.07	0.00	0.03	0.25	0.34	0.00	0.00	0.08	0.31	0.00	0.63	0.00	0.36	0.89	0.68	0.02	0.66	0.49	0.88	0.16	0.45
left_wrist_y	0.23	0.00	0.52	0.00	0.17	0.00	0.01	0.00	0.32	0.00	0.00	0.00	0.35	0.00	0.13	0.00	0.26	0.00	0.46	0.00	0.64	0.00	0.73	0.35	0.19	0.32
right_wrist_x	0.00	0.05	0.37	0.06	0.00	0.22	0.93	0.87	0.00	0.08	0.08	0.35	0.00	0.01	0.00	0.86	0.01	0.43	0.00	0.91	0.01	0.00	0.00	0.21	0.01	0.47
right_wrist_y	0.52	0.00	0.73	0.00	0.29	0.00	0.78	0.02	0.01	0.10	0.31	0.00	0.01	0.00	0.13	0.00	0.66	0.00	0.00	0.00	0.03	0.06	0.08	0.99	0.02	0.77
left_hip_x	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.02	0.00	0.53	0.00	0.13	0.00	0.13	0.00	0.33	0.00	0.49	0.04	0.73	0.16	0.53	0.34	0.96	0.28	0.64
left_hip_y	0.60	0.00	0.88	0.00	0.50	0.00	0.80	0.00	0.83	0.00	0.63	0.00	0.86	0.00	0.33	0.00	0.26	0.00	0.73	0.00	0.48	0.00	0.65	0.47	0.32	0.06
right_hip_x	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.07	0.00	0.26	0.01	0.66	0.00	0.26	0.00	0.27	0.55	0.42	0.01	0.85	0.99	0.48	0.06	0.79
right_hip_y	0.78	0.00	0.40	0.00	0.97	0.00	0.25	0.00	0.38	0.00	0.36	0.00	0.43	0.00	0.49	0.00	0.27	0.00	0.33	0.00	0.62	0.00	0.50	0.78	0.94	0.17
left_knee_x	0.03	0.15	0.23	0.19	0.00	0.50	0.05	0.76	0.00	0.06	0.89	0.46	0.00	0.00	0.04	0.73	0.55	0.33	0.00	0.45	0.00	0.00	0.00	0.11	0.00	0.15
left_knee_y	0.21	0.00	0.13	0.00	0.32	0.00	0.75	0.01	0.94	0.02	0.68	0.00	0.91	0.00	0.73	0.00	0.42	0.00	0.45	0.00	0.06	0.00	0.84	0.61	0.13	0.64
right_knee_x	0.28	0.91	0.00	0.77	0.48	0.71	0.00	0.39	0.00	0.06	0.02	0.64	0.01	0.03	0.16	0.48	0.01	0.62	0.00	0.06	0.00	0.06	0.00	0.09	0.00	0.03
right_knee_y	0.33	0.00	0.52	0.00	0.14	0.00	0.20	0.00	0.00	0.00	0.66	0.00	0.00	0.06	0.53	0.00	0.85	0.00	0.00	0.00	0.06	0.00	0.01	0.95	0.26	0.44
left_ankle_x	0.09	0.22	0.05	0.41	0.00	0.56	0.00	0.83	0.00	0.14	0.49	0.73	0.00	0.08	0.34	0.65	0.99	0.50	0.00	0.84	0.00	0.01	0.00	0.00	0.00	0.00
left_ankle_y	0.06	0.21	0.32	0.43	0.00	0.13	0.12	0.27	0.09	0.11	0.88	0.35	0.21	0.99	0.96	0.47	0.48	0.78	0.11	0.61	0.09	0.95	0.00	0.00	0.00	0.00
right_ankle_x	0.78	0.41	0.00	0.79	0.10	0.52	0.00	0.82	0.00	0.55	0.16	0.19	0.01	0.02	0.28	0.32	0.06	0.94	0.00	0.13	0.00	0.26	0.00	0.00	0.00	0.00
right_ankle_y	0.28	0.07	0.11	0.16	0.01	0.04	0.02	0.06	0.22	0.03	0.45	0.32	0.47	0.77	0.64	0.06	0.79	0.17	0.15	0.64	0.03	0.44	0.00	0.00	0.00	0.00

Figure 2: p-values of Spearman rank correlation between all key points.

By examining the overlap between these tables, it can be determined which coordinate pairs can most effectively be implemented into a machine learning model to form an accurate run. It is important to note that when a variable is paired against itself (nose_x against nose_x, nose_y against nose_y, etc.), it will always produce a p-value of 0 and a correlation coefficient of

1. These pairings are a side effect of matching all of the variables in this way, and they do not represent any meaningful correlations. The tables are combined in Figure 3, in which a value of 1 represents a coordinate pair with a correlation coefficient of 0.6 or higher and a p-value below 0.05, while a value of 0 represents a coordinate pair that does not meet these conditions, thus deeming the correlation insignificant.

Key Points	nose_x	nose_y	left_shoulder_x	left_shoulder_y	right_shoulder_x	right_shoulder_y	left_elbow_x	left_elbow_y	right_elbow_x	right_elbow_y	left_wrist_x	left_wrist_y	right_wrist_x	right_wrist_y	left_hip_x	left_hip_y	right_hip_x	right_hip_y	left_knee_x	left_knee_y	right_knee_x	right_knee_y	left_ankle_x	left_ankle_y	right_ankle_x	right_ankle_y
nose_x	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
nose_y	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
left_shoulder_x	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0
left_shoulder_y	0	1	0	1	0	1	0	1	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
right_shoulder_x	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0
right_shoulder_y	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	0
left_elbow_x	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
left_elbow_y	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0
right_elbow_x	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0
right_elbow_y	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0
left_wrist_x	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
left_wrist_y	0	0	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
right_wrist_x	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0
right_wrist_y	0	1	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0
left_hip_x	1	0	1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
left_hip_y	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0
right_hip_x	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
right_hip_y	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0
left_knee_x	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0
left_knee_y	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0
right_knee_x	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0
right_knee_y	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0
left_ankle_x	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0
left_ankle_y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
right_ankle_x	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1
right_ankle_y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 3: p-values of Spearman rank correlation between all key points.

To consider the run modelable through machine learning, each coordinate should be moderately correlated with a minimum of at least one other coordinate through the passage of time. The machine learning model used in the study was a deep neural network. The form of this

type of neural network allows it to make predictions based on the learned connections between variables, as discussed by Cser et al. Keeping in mind the symmetry that exists in the table, it can be observed that a moderate to strong correlation exists between each single coordinate and at least one other coordinate. This indicates that despite the unpredictability of humans and their motion, a consistent top level run can be modeled to a limited extent with a deep neural network.

Limitations

Creating the model utilizing the proposed structure (a deep neural network with 5 hidden layers of 512 parameters each) uncovers a slew of limitations that are necessary to address in its evaluation. After evaluating the final model using the model.evaluate() function in the Keras library, it is revealed that the model predicts the next frame with a loss of 18.125 and an accuracy of 66.67 percent. This result is within expectation, as the correlation coefficient of at least one coordinate pair including every tested coordinate was greater than or equal to 0.6, meaning that the accuracy of an ML model modeling this relationship should optimally have an accuracy that falls between 0.6 and 1. Although this accuracy is acceptable when making predictions one frame at a time, attempting to make predictions more than a single frame into the future results in the inaccuracy compounding, decreasing drastically as the number of frames increases. The proposed prediction of 15 frames into the future for a reliable model can only be completed with an accuracy of just 0.228 percent.

Stemming from human unpredictability, this compounding accuracy is the ultimate limiting factor in allowing the model to predict further than a frame into the future as proposed. However, a side effect of the way that the model was fitted is its ability to correct the posture/form of any running pose. Because the data used to fit the model was retrieved from an

Olympic running event, predicted poses will be aligned with the style of the Olympic runner, a favorable position for any runner below the Olympic level. This means that rather than trying to predict 15 frames at once (essentially generating a full corrected stride based on one starting position), a runner's position can be corrected on each individual frame, retaining a 66.67 percent accuracy through the entire run.

This solution is not perfect, however, as discontinuity is introduced as a problem. When correcting the pose on a certain frame, the model has no context beyond the singular frame that is inputted. The model cannot see where each point came from, so values such as velocity and acceleration are not considered individually for each runner. Rather, all inference is based on values that were observed when the model was first fitted. This, along with the model's accuracy, causes the model to correct each frame individually, leading to a complete run that appears to jitter due to inconsistency when played back (each frame introduces its own inaccuracy unrelated to other frames). Depending on the use case, correcting a run with continuity may not be necessary. Finding the least accurate point in the run, for example, can be done without continuity by calculating the cosine similarity between expected and predicted pose at each frame. For continuity to be introduced, the model would have to be recreated with an ability to input contextual poses.

Conclusion and Future Directions

Evaluating the results of the creation of the deep learning model against Wei's observations reveals the placement of a deep learning and computer vision approach to running analysis among modern, widespread sports technologies. The proposed method has a strong resemblance of the method used by MySwing to aid golfers in golf swing analysis. Both utilize

pose detection to visually show an athlete what they should change about their form in order to improve. Using MySwing, “players can observe and analyze their own swings from a comprehensive perspective and compare them with more skilled professional golfers" (Wei, et al., 2021). However, the proposed method benefits from its ability to perform pose detection without the need for any external hardware beyond a camera, one of MySwing’s greatest downfalls. This practicality leads to a solution that is easily accessible to anyone with just a smartphone, increasing the potential widespread adoption. This adoption would allow athletes without access to coaching due to financial or other constraints to further develop their running form. Casual runners would benefit from the adoption of this technology as well, as these types of runners may not invest in coaching. This technology would allow these athletes to better avoid injuries by helping them correct faulty, possibly dangerous, form. Heightening the performance potential for all runners has the advantage of making the sport more competitive as more athletes gain access to the tools they need to reach a high performance level in the sport. A higher level of competition overall makes the sport more enjoyable for athletes and fans alike.

Two distinct conclusions can be made after analyzing the results of the data collection. The first conclusion is in regard to the ability of a run to be modeled through deep learning. This study analyzes one particular Olympic runner’s stride, proving that a correlation strength of 0.6 or higher can be observed between each component of a pose and at least one other component. This result demonstrates that a run can be modeled with an ideal accuracy greater than 60 percent, which can be seen in the implementation used in this study. Making corrections with this level of accuracy provides evidence that this technology does have potential widespread appeal to runners.

In a study by Paul Fleming, interviews reveal that one of the six primary themes that runners and coaches expect technology to be able to address is technique. The sub-themes of technique addressed by deep learning are kinematics and joint position/angle. Thus, the second conclusion that can be made is that the output of the suggested deep learning approach to the problem of running does align with the information required by runners and coaches to make informed decisions about how their running can be improved. However, the conclusion is limited because the model does not address the multitude of other factors that were brought up in Fleming's study. Therefore, a machine learning approach paired with pose estimation is best suited as a supplement to many of the existing training tools that exist, including wearables as brought up in Borowski-Beszta and Polasik's research.

Although the model presented in this research has a limited accuracy and thereby a limited scope of application, it seeks to set the groundwork for future models to be created that address more of the aspects of a run. There are various things that should be done in the future to advance the solution presented in this research and address the apparent limitations. The current model is limited to a single perspective, a side view in which the runner is stationary within the frame. This can be expanded up by increasing the size of the dataset and the variety of data included. The model created in this study is based on the stride of one specific olympic runner, a feasible method for a run contained within one perspective or simply as a proof of concept. However, in order to appeal to a wider audience, a dataset must be taken from a set of professional runners, runs recorded from a variety of angles. Doing this is far more computationally expensive than was feasible for this study, but would be fitting for a larger organization or institution to tackle. Furthermore, the limited dataset used prevents the model

from accurately correcting the posture of runners with significantly different body proportions from the runner that the model was trained on.

The results of this study set the framework for a new category of technology suited for supplementing the coaching of runners. The research has significance in the field of running as a simpler way to apply more complex analysis to a run. The method avoids the use of bulky sensors that are typically used when analyzing a run, while having a high potential for new insight. The ease of use of an analytical running technology that is free from the burden of physical sensors that inhibit motion is the principle behind the deep learning model created in this research. With future advancements, one can expect the technology to expand in its capabilities, allowing it to target more of the themes presented by Fleming. Additionally, this technology isn't limited to running. Form, specifically involving body position, is an important aspect in a majority of sports. The process utilized in this research could feasibly be replicated with almost any sport with very little modification, a further benefit of the simplicity of the technology in comparison with others and the versatility of neural networks. Beyond this, the rapid advancement of Large Language Models (text generation models with an understanding of natural language) means that the running model proposed in this research could be adapted to give written feedback, including specific details about how to achieve a specific result rather than just displaying the expected result. This could involve training plans, specific pointers, and explanations of the theory behind why posture should look a certain way.

References

Borowski-Beszta, M., & Polasik, M. (2020). Wearable devices: New quality in sports and finance. *Journal of Physical Education and Sport*, *20*, 1077–1084. https://doi.org/10.7752/jpes.2020.s2150

Cser, J., Oosterlaan, L. M., van der Veer, P., Heemink, A. W., & van de Graaf, J. (2001). A Neural Network Approach to Predict Nearshore Morphology along the Dutch Coast. *Journal of Coastal Research*, 143–153.

Fleming, P., Young, C., Dixon, S., & Carré, M. (2010). Athlete and coach perceptions of technology needs for evaluating running performance. *Sports Engineering*, *13*(1), 1–18. https://doi.org/10.1007/s12283-010-0049-9

Krogh, A. (2008). What are artificial neural networks?. Nature biotechnology, 26(2), 195-197.

Lv, J., Jiang, X., & Jiang, A. (2022). Application of Virtual Reality Technology Based on Artificial Intelligence in Sports Skill Training. *Wireless Communications & Mobile Computing*, 1–7. https://doi.org/10.1155/2022/4613178

Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., Bileschi, S., Terry, M., Nicholson, C., Gupta, S. N., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viégas, F. B., & Wattenberg, M. (n.d.). *TensorFlow.js: Machine Learning for the Web and Beyond*. 13.

Song, Y., Schwing, A., & Urtasun, R. (2016, June). Training deep neural networks via direct loss minimization. In International conference on machine learning (pp. 2169-2177). PMLR.

Wei, S., Huang, P., Li, R., Liu, Z., Zou, Y., & Link to external site, this link will open in a new window. (2021). Exploring the Application of Artificial Intelligence in Sports Training: A Case Study Approach. *Complexity*, *2021*. https://doi.org/10.1155/2021/4658937

Yiannakides, A., Aristidou, A., & Chrysanthou, Y. (2019). Real‐time 3D human pose and motion reconstruction from monocular RGB videos. *Computer Animation and Virtual Worlds*, *30*(3–4). https://doi.org/10.1002/cav.1887

Zhou, Q., Wang, J., Wu, P., & Qi, Y. (2021). Application Development of Dance Pose Recognition Based on Embedded Artificial Intelligence Equipment. *Journal of Physics: Conference Series*, *1757*(1), 012011. https://doi.org/10.1088/1742-6596/1757/1/012011

Appendix A

Pose Detection Code

Code utilizing BlazePose to detect a runner's key points in the Javascript programming language.

```
import * as poseDetection from "@tensorflow-models/pose-detection";
import "@tensorflow/tfjs-backend-cpu";
import "@tensorflow/tfjs";
import * as mpPose from "@mediapipe/pose";
import { data } from "@tensorflow/tfjs";

let ended = false;
let allposes = [];

// sleep helper function

const sleep = (ms) => {
  return new Promise((res) => {
    setTimeout(() => {
      res(true);
    }, ms);
  });
};

// main pose detection function

let processor = {
  frameNumber: 0,
  canStep: true,
  hasLoaded: false,
  timerCallback: async function () {
    let self = this;
    if (self.frameNumber > self.videoTotalFrames && !ended) {
      ended = true;
      let formattedposes = [];
      allposes.map((e) => {
        let keypointsar = [];
        e[0].keypoints.map((p) => {
          keypointsar.push(p.x);
          keypointsar.push(p.y);
        });
        formattedposes.push([keypointsar.join(",")]);
      });
      let csvContent =
```

```
        "data:text/csv;charset=utf-8," +
        formattedposes.map((e) => e.join(",")).join("\n");
      var encodedUri = encodeURI(csvContent);
      var link = document.createElement("a");
      link.setAttribute("href", encodedUri);
      link.setAttribute(
        "download",
        `${document.getElementById("video").src}.csv`
      );
      document.body.appendChild(link);

      link.click();
      return;
    }
    if (this.canStep) {
      console.log(`Frame: ${self.frameNumber}`);
      await this.computeFrame();
      self.frameNumber++;
      setTimeout(function () {
        this.video.currentTime = Math.max(0, self.frameNumber / 10);
        self.timerCallback();
      }, 0);
    } else {
      setTimeout(function () {
        self.timerCallback();
      }, 100);
    }
  },

  doLoad: function () {
    if (!this.hasLoaded) {
      this.hasLoaded = true;
      this.video = document.getElementById("video");
      this.video.addEventListener("ended", () => {
        ended = true;
      });
      this.videoTotalFrames = Math.max(0, this.video.duration * 10);
      this.video.crossOrigin = "anonymous";
      this.c1 = document.getElementById("c1");
      this.c1.height = this.video.videoHeight / 2;
      this.c1.width = this.video.videoWidth / 2;
      this.ctx1 = this.c1.getContext("2d");
      this.c2 = document.getElementById("c2");
      this.c2.height = this.video.videoHeight / 2;
      this.c2.width = this.video.videoWidth / 2;
      this.ctx2 = this.c2.getContext("2d");
      let self = this;
```

```
      this.video.currentTime = 0;
      this.video.addEventListener(
        "play",
        function () {
          self.video.pause();
          self.width = self.video.videoWidth / 2;
          self.height = self.video.videoHeight / 2;
          self.timerCallback();
        },
        false
      );
    }
  },

  computeFrame: async function () {
    this.canStep = false;
    await this.ctx1.drawImage(this.video, 0, 0, this.width, this.height);
    let frame = this.ctx1.getImageData(0, 0, this.width, this.height);
    let image = new Image();
    this.ctx1.putImageData(frame, 0, 0);
    image.src = this.c1.toDataURL();
    const model = poseDetection.SupportedModels.BlazePose;
    const detector = await poseDetection.createDetector(model, {
      runtime: "tfjs",
    });
    const poses = await detector.estimatePoses(image);
    allposes.push(poses);

    // display skeleton for debugging

    this.ctx2.putImageData(frame, 0, 0);
    this.ctx2.fillStyle = "red";
    for (let i = 0; i < 33; i++) {
      if ([13, 14].includes(i)) {
        continue;
      }
      let x = poses[0].keypoints[i].x;
      let y = poses[0].keypoints[i].y;
      this.ctx2.beginPath();
      this.ctx2.arc(x, y, 5, 0, 2 * Math.PI);
      this.ctx2.fill();
    }
    this.ctx2.beginPath();
    this.ctx2.strokeStyle = "blue";
    this.ctx2.lineWidth = 6;

    // draw line segments to display
```

```
    connectpoints(25, 27, this.ctx2, poses);
    connectpoints(23, 25, this.ctx2, poses);
    connectpoints(24, 23, this.ctx2, poses);
    connectpoints(11, 12, this.ctx2, poses);
    connectpoints(24, 26, this.ctx2, poses);
    connectpoints(26, 28, this.ctx2, poses);
    connectpoints(12, 14, this.ctx2, poses);
    connectpoints(14, 16, this.ctx2, poses);
    connectpoints(24, 12, this.ctx2, poses);
    connectpoints(23, 11, this.ctx2, poses);
    this.ctx2.stroke();
    this.canStep = true;
    return;
  },
};

// pose display helper function

const connectpoints = (p1, p2, ctx, poses) => {
  ctx.moveTo(poses[0].keypoints[p1].x, poses[0].keypoints[p1].y);
  ctx.lineTo(poses[0].keypoints[p2].x, poses[0].keypoints[p2].y);
};

window.onload = () => {
  processor.doLoad();
};
```

Appendix B

Machine Learning Model Code

Code utilizing Tensorflow to create and train the running model in the Python programming language.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import cv2 as cv
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

column_names = ['nosex', 'nosey', 'leix', 'leiy', 'lex', 'ley', 'leox', 'leoy',
'reix', 'reiy', 'rex', 'rey', 'reox', 'reoy', 'left_earx', 'left_eary',
'right_earx', 'right_eary', 'mlx', 'mly', 'mrx', 'mry', 'left_shoulderx',
'left_shouldery', 'right_shoulderx', 'right_shouldery', 'left_elbowx',
'left_elbowy', 'right_elbowx', 'right_elbowy', 'left_wristx', 'left_wristy',
'right_wristx', 'right_wristy', 'lpx', 'lpy', 'rpx', 'rpy', 'lix', 'liy', 'rix',
'riy', 'ltx', 'lty', 'rtx', 'rty', 'left_hipx', 'left_hipy', 'right_hipx',
'right_hipy', 'left_kneex', 'left_kneey', 'right_kneex', 'right_kneey',
'left_anklex', 'left_ankley', 'right_anklex', 'right_ankley', 'lhx', 'lhy', 'rhx',
'rhy', 'lfix', 'lfiy', 'rfix', 'rfiy']
important_column_names = ['nosex', 'nosey', 'left_shoulderx', 'left_shouldery',
'right_shoulderx', 'right_shouldery', 'left_elbowx', 'left_elbowy', 'right_elbowx',
'right_elbowy', 'left_wristx', 'left_wristy', 'right_wristx', 'right_wristy',
'left_hipx', 'left_hipy', 'right_hipx', 'right_hipy', 'left_kneex', 'left_kneey',
'right_kneex', 'right_kneey', 'left_anklex', 'left_ankley', 'right_anklex',
'right_ankley']

raw_dataset = pd.read_csv("./csvs/run.csv", names=column_names,
                          na_values='?', comment='\t',
                          sep=',', skipinitialspace=True)
dataset = raw_dataset.copy()
dataset.dropna()
dataset = dataset.drop(columns=['leix', 'leiy', 'lex', 'ley', 'leox', 'leoy',
'reix', 'reiy', 'rex', 'rey', 'reox', 'reoy', 'left_earx', 'left_eary',
'right_earx', 'right_eary', 'mlx', 'mly', 'mrx', 'mry', 'lpx', 'lpy', 'rpx', 'rpy',
'lix', 'liy', 'rix', 'riy', 'ltx', 'lty', 'rtx', 'rty', 'lhx', 'lhy', 'rhx', 'rhy',
'lfix', 'lfiy', 'rfix', 'rfiy'])
train_dataset = dataset.sample(frac=0.8)
```

```
train_dataset = dataset[:25]
test_dataset = dataset.drop(train_dataset.index)
test_dataset = dataset[25:]
train_features = train_dataset.copy().drop([24])
test_features = test_dataset.copy().drop([31])

train_labels = train_dataset.copy().drop(0)
test_labels = test_dataset.copy().drop(25)

normalizer = tf.keras.layers.Normalization(axis=-1)
normalizer.adapt(np.array(train_features))

first = np.array(train_features[:1])
run_model = tf.keras.Sequential([
    normalizer,
    layers.Dense(units=512, activation='relu'),
    layers.Dense(units=512, activation='relu'),
    layers.Dense(units=512, activation='relu'),
    layers.Dense(units=512, activation='relu'),
    layers.Dense(units=512, activation='relu'),
    layers.Dense(units=26)
])

run_model.compile(
    optimizer=tf.keras.optimizers.legacy.Adam(0.01),
    loss='mean_absolute_error', metrics='accuracy')

import datetime

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=1)

# train model
history = run_model.fit(
    train_features,
    train_labels,
    epochs=500,
    # Calculate validation results on 40% of the training data.
    validation_split = 0.4,
    callbacks=[tensorboard_callback])

hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch

loss, acc = run_model.evaluate(test_features, test_labels)
print("Loss {}, Accuracy {}".format(loss, acc))
```

```
# make predictions
predictindex = 1 # frame to predict

# plot predictions and actual values as bar chart
def plot_error_bar():
  for i in range(len(np.array(test_features)[0])):
    plt.bar(3*i - 1, np.array(test_features)[0][i], color="blue")
  for i in range(len(pose_predictions[0])):
    plt.bar(3*i, pose_predictions[0][i], color="red")
  plt.ylim(0, 300)
  plt.xlabel('pose_component')
  plt.ylabel('positions')
  plt.legend()
  plt.grid(True)

# plot predictions and actual values as line graph
def plot_error_line():
  plt.plot(np.array(test_features[predictindex-1:predictindex])[0],
label='correct')
  plt.plot(pose_predictions[0], label='guess')
  plt.ylim(0, 300)
  plt.xlabel('pose_component')
  plt.ylabel('positions')
  plt.legend()
  plt.grid(True)

plot_error_bar()
plot_error_line()
```