Using open-source deep learning tools for studying biomechanics

Pawel Kudzia
Simon Fraser University

Literature review

With significant advancements in deep learning for computer vision, the feasibility of acquiring large-scale behavioural data from video has become more realistic. One of the significant hurdles in extracting biomechanical data from video is the cumbersome and time-consuming nature of manually labelling anatomical landmarks, or key-points, in video frames [1], [2]. The tedious nature of manually locating landmarks and training personnel is impractical and is prone to error deterring researchers from video datasets. The ability to use deep neural networks to identify anatomical landmarks allows for computer vision applications that reduce the time constraints typically associated with video analysis. Such deep learning tools increase the potential applications in the field, where collecting data outside of the constrained lab environment, more typical or real-life behaviour, is made possible [3]. With the wide open-source availability of deep learning frameworks for image recognition tasks, there is also an opportunity to inexpensively integrate such tools for educational purposes in the engineering classroom. My overall goal in this project was to use open source tools to train and test a neural network for identifying anatomical landmarks in videos of people vertically jumping and then to use this data as a proof of concept to evaluate biomechanical outputs.

Human pose estimation is a visual recognition task dealing with the autonomous localization of anatomical landmarks in the frame by frame analysis of video data. The common approach to identifying anatomical landmarks is by physically placing reflective markers and a marker-based motion capture camera in an instrumented lab-based environment captures the movement of these markers [4]. Although marker-based systems are widely used, this approach is limiting data collection to those who have specialized tools, to indoor research facilities, and tracking of only the features where the markers have been placed [3], [4]. To study the behaviours of wild animals or human movement in settings outside of the lab, video data provides a low level of invasiveness and cost. Autonomous detection of human and animal poses from the video has grown over the past decade where neural networks and deep learning tools are used for visual recognition and pose estimation. Work such as DeepPose [5], DeeperCut [6], OpenPose [7], and DeepLabCut [3], [8] use deep learning to make predictions of relevant anatomical features in videos. Of these tools, DeepLabCut is an open-source and freely available tool that integrates easily with Python. DeepLabCut takes advantage of an already pre-trained neural network architecture based on the ImageNet database (~14 million images), a massive dataset for object recognition [9], to provide feasible and efficient tools that scale for use in research and education.

One of the central features of the DeepLabCut framework is the use of transfer learning. To achieve high accuracy in anatomical landmarking tasks neural networks have to be trained using large amounts of labelled training data (for example, ~25,000 frames in the MPII Human Pose dataset [10]) using very deep network architectures such as the Residual Network 50 (ResNet-50) or Residual Network 101 (ResNet-101) [11]. However, the best performing algorithms from pose estimation of humans and animals use deep features [5], it then perhaps not surprising that deeper neural network architectures result in improved accuracy [3]. The issue is that deep architectures require a tremendous amount of training data and training time to achieve a high level of accuracy and reliability [6], [12]. In real-world applications, the typical user will ideally want to define the location of a handful of anatomical landmarks by only manually labelling a few hundred frames on a small subset of data and hoping it generalizes well on novel video data [13]. To make this possible, DeepLabCut uses 1) a pre-trained ResNet model and 2) deconvolutional layers [3]. First, the ResNet network is populated with the training weights determined when trained on the ImageNet database, on which it achieves excellent performance. Now the classification layer at the output of the ResNet model is removed and deconvolutional layers are used to up-sample the visual information and produce spatial probability densities [3], [6]. This idea is based on the pose estimation and object recognition algorithm suggested by DeeperCut, but the general idea here is that for each body segment its probability density represents the evidence that a body part is in a particular location. To fine-tune the weights of the deconvolutional layers (each layer represents a landmark) for a particular image recognition task, its weights are trained on the small sample of user-defined labelled data (this can be based on what the user is interested in in a specific dataset). Here during training, the weights are adjusted in an iterative fashion such that for a given frame the network assigns high probabilities to labelled body part locations and low probabilities elsewhere. This type of rewiring of the pre-trained ResNet neural network architecture, trained on the massive dataset taken from ImageNet, results in a highly robust and efficient tool requiring minimal time to train (**Figure 1**).

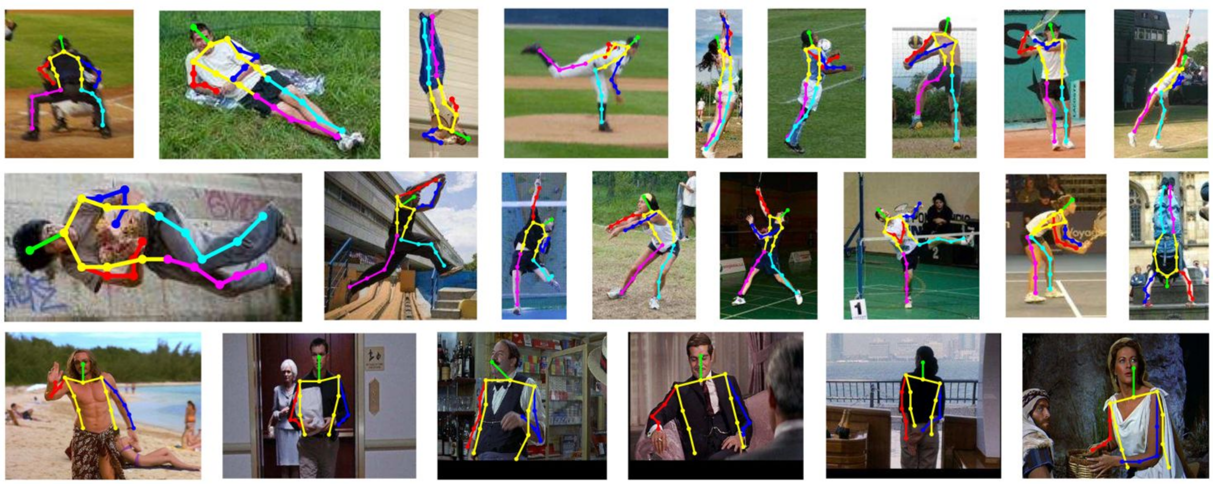

Figure 1: A series of examples of human pose / human joint position estimations using predictions from deep neural networks on video datasets of individuals in team sports and daily living scenarios (based on [6]).

In this project, my overall goal was to use the DeepLabCut to train and test a neural network for identifying useful anatomical landmarks in videos of people vertically jumping and then to use this data to evaluate biomechanical outputs. In addition to this main goal, I had a secondary goal of providing examples of how these types of tools could be used in the classroom to educate students on using deep learning for studying biomechanics. To accomplish these goals, I had four specific aims. My first aim was to collect video data of people jumping to be used to train and evaluate a neural network in identifying 6 anatomical landmarks. To accomplish this aim, I recruited 13 participants using social media and through word of mouth. Interested participants sent me several videos each, recorded using their phones, of themselves jumping. I then used DeepLabCut, an open-source Python tool, to train and evaluate a neural network to identify anatomical landmarks on the participant's bodies. My second aim was to determine sagittal plane joint angles from the predicted anatomical landmarks. To accomplish this aim, I used the trained neural network to predict anatomical landmarks of novel jumping videos. I used the positions of the predicted landmarks to perform inverse dynamics and estimate sagittal plane leg kinematics. My third aim was to use the trained neural network and determine how well it generalizes on novel videos on participants the network was not trained on. To do this, I used the 3 participants that the neural network has never seen before and evaluated how well it predicted anatomical landmarks. Finally, my fourth goal was to provide an example of how this type of tool could be used in the classroom. To accomplish this goal, I wrote code to provide visualization of the RGB video, predicted landmarks, and joint angle estimation.

Methods

Participant Recruitment

I recruited 13 participants for this study. I recruited participants online using Twitter and through word of mouth. When a participant expressed interest, I explained to them the goals of the study and the data I would be collecting. All participants agreed to these terms and provided me with several videos of themselves jumping. Each video contained a single jump, recorded in landscape mode (1280x720 pixels). I instructed participants to perform several jumps while the video was recorded from the sagittal plane. I told them they could do this either with the help of another person or by simply leaning the camera against a wall. My goal was to keep data collection as simple as possible as to not limit participation and minimize the need for assistance.

Data Curation

I converted each video from each participant into a .mp4 file format and trimmed it as needed. I used iMovie to trim each jumping video such that all the videos contained only the jumping phases of the jump. I trimmed each video to start at the point where the participant began their jump and ended it immediately after they reached a standing posture after the landing from their jump. Next, I randomly chose 10 participants to be a part of my training and testing dataset and 3 participants to be part of my generalization data set.

Neural Network

I trained a neural network to predict anatomical landmarks from the video frames. First, I used the open-source package DeepLabCut [3], [8], [14] to extract RGB frames from the videos. Here I used one video from each of the 10 participants that were part of the training/testing dataset and using DeepLabCut extracted 20 frames from each video and each participant. The algorithm in DeepLabCut scans the video and selects representative frames throughout the video that provide variation between frames. Next, using the DeepLabCut interface, I manually labelled anatomical landmarks in each frame. The landmarks that I marked were the ear, the armpit, the hip, the knee, the ankle, and the end of the foot. The ear was chosen as it was visible throughout all images making it amenable for tracking. Next, the armpit was used as an approximate location of the centre of mass of the participant and again it was amenable for tracking. I chose the hip, knee, ankle, and end of the foot, to model the leg. When the landmark was obstructed or difficult to identify accurately, I used my intuition to make an approximation. I chose these landmarks intending to perform inverse dynamic analysis of the leg and estimate sagittal plane leg angles.

I used a preconstructed neural network architecture for training. The open-source package DeepLabCut provides several options for choosing the neural network architecture such as ResNet-50, ResNet-101 and ResNet-152 [3], [8]. Here, I chose to use the ResNet-50 architecture for this application as this 50 layer convolutional neural network should be fully adequate for the task with a single participant in each video data frame. If multiple participants were jumping in one frame, which is a much more challenging problem to solve for labelling, it is recommended to use a larger architecture such as the Resnet-101 [3]. I trained the neural network on the free cloud-based server Google Colab, providing me with access to their online GPU [15]. Training this network on my laptop would take between 10-20x longer given the large number of iterations that were recommended [8]. I wrote Python code in a Jupyter notebook following the proposed guidelines as indicated in the DeepLabCut GitHub repository [16]. I evaluated the performance of three different datasets as shown in **Table 1**. The dataset was split 95% training and 5% evaluation. The objective function of the neural network was to minimize the loss between predicted and manually labelled landmarks. The loss was measured by computing the mean average Euclidean error which is proportional to the average root mean square error between the manual labels and the ones predicted by DeepLabCut [8]. I used a batch size of 1 for all training.

Table 1: Neural network specification and epochs evaluated for ResNet-50

	Set 1	Set 2	Set 3
Manually labelled frames (1 participant = 20 frames)	100	160	200
Epochs	10000	57000	129000
Processing time in GoogleColab using GPU	~1.5 hours	~ 8 hours	~ 18 hours
Batch size	1	1	1

After training, I evaluated the performance of the network on participants the network was trained on and on novel videos it has never seen before. First, I evaluated the performance of the network in predicting anatomical landmarks on participants it was trained on. I used a different video that was used in the trained set. Here I evaluated performance by recording the estimated likelihood ratios of the network's predictions and by plotting the predictions overtop of the video to visually assess performance. DeepLabCut provides python functions that output the predicted landmarks overlaid onto the video. To test generalization, I evaluated the performance of the network in predicting anatomical landmarks on participants that the network was not trained on. Here

Inverse Dynamics

To determine the angles between the joints I use the predicted 2D positions of each of the anatomical landmarks. Each anatomical landmark prediction has an (x,y) coordinate. I provide the code for this in the supplementary section but in brief, this requires calculating the segment vectors between the points which tells me the orientation of each segment relative to one of the joints. Next, to determine the segment angle, I begin by calculating the segment unit vector which provides me with a vector that points from one segment endpoint (for example the ankle marker) to the other (for example the knee marker). Finally, I take the dot product of the segment unit vectors to find the angle between the two vectors. For presentation in the results, I subtract the joint angles of the person standing still at the beginning of the jump. This results in the joint angles starting at either increasing or decreasing according to the convection presented in **Figure 2**. For comparison between participants, I shift the joint angles to align based on the peak vertical height achieved. At this instance, participants will have maximum plantarflexion.

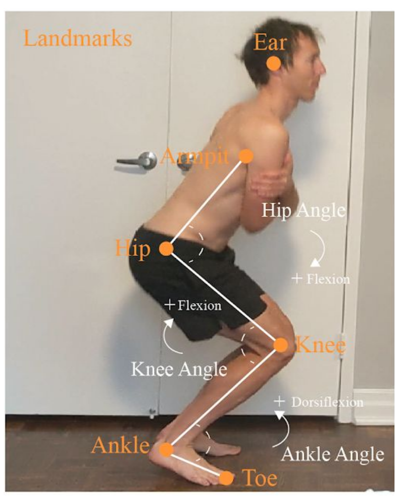

Figure 2: Once the (x,y) position of the landmark was predicted I evaluated the leg joint angles according to the conventions shown here.

Results

The neural network well predicted anatomical landmarks

I found that a larger training dataset with more training iterations resulted in better predictions. When training the neural network with a smaller dataset and fewer iterations (Set 1, 5 participants in the training set) I found that this resulted in the worst performance (**Figure 3**). When I increased the training data set to 10 participants containing 200 manually labelled frames, I found that increases in the number of iterations only resulted in marginal performance increases (Set 2 vs Set 3 in **Figure 3**). I found that the final loss function values for Set 2 were .0036 with a training pixel error of 3.74 pixels and a validation error of 7.92 pixels. Compared to Set 3 the final loss function value was 0.0026 with a training pixel error of 3.00 pixels and a validation error of 7.36 pixels. This can be compared to the typical human labelling error that has been reported for 800 × 800-pixel-sized datasets whose human-level accuracy was 2.7 pixels with 500 labelled frames [8].

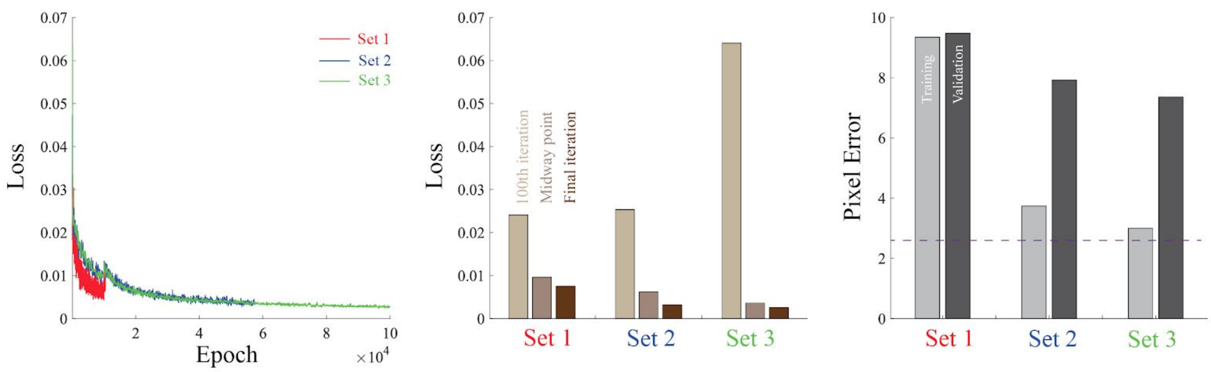

Figure 3: Results for evaluated neural network results for Set 1, Set 2, and Set 30 **Left:** The value of the loss function as the number of training iterations increases. **Center:** The value of the loss function at three different instances during training. At the 100th epoch, the midway point, and the final epoch. **Right:** The pixel error for the training and validation for the three sets. The horizontal line here indicates the reported pixel error for manual labelling [8]**.**

I found that the predicted anatomical landmarks had high likelihoods across all participants. In general, the neural network predicted anatomical landmarks with high probabilities close to 1 for all 5 landmarks across all participants (**Figure 4**). Overall, the medium probability for all 5 landmarks was >0.98. Upon closer inspection of the frames that contained lower probabilities of <0.5 in these cases, the participants were wearing baggier clothing that shifted when they jumped or they wore long black pants that resulted in poor contrast between their legs and the background. A representative visual diagram is provided in **Figure 5,** here I show 3 participants comparing the manually labelled landmark and the neural network predicted landmarks.

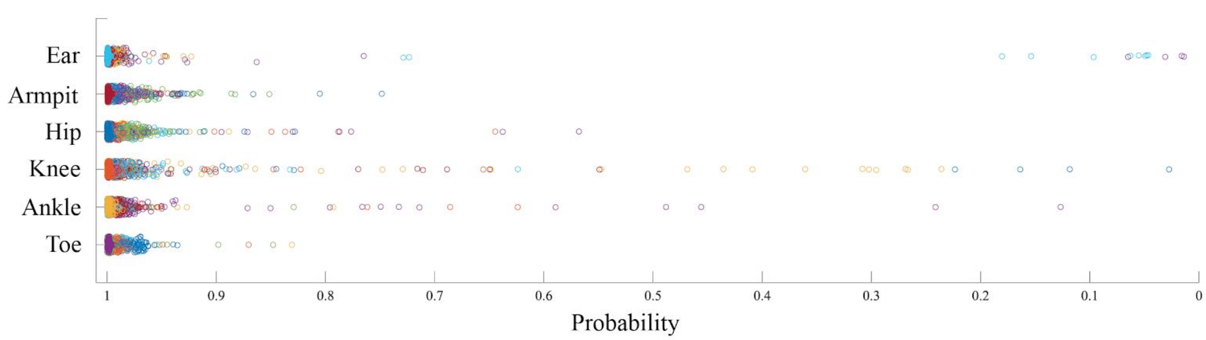

Figure 4: The determined probabilities of the neural network predictions for the 6 anatomical landmarks. Each dot is the probability from a single frame of data. Dots of different colours represent different participants. A probability of 1 would mean 100% confidence in the prediction of the neural network of the landmarks.

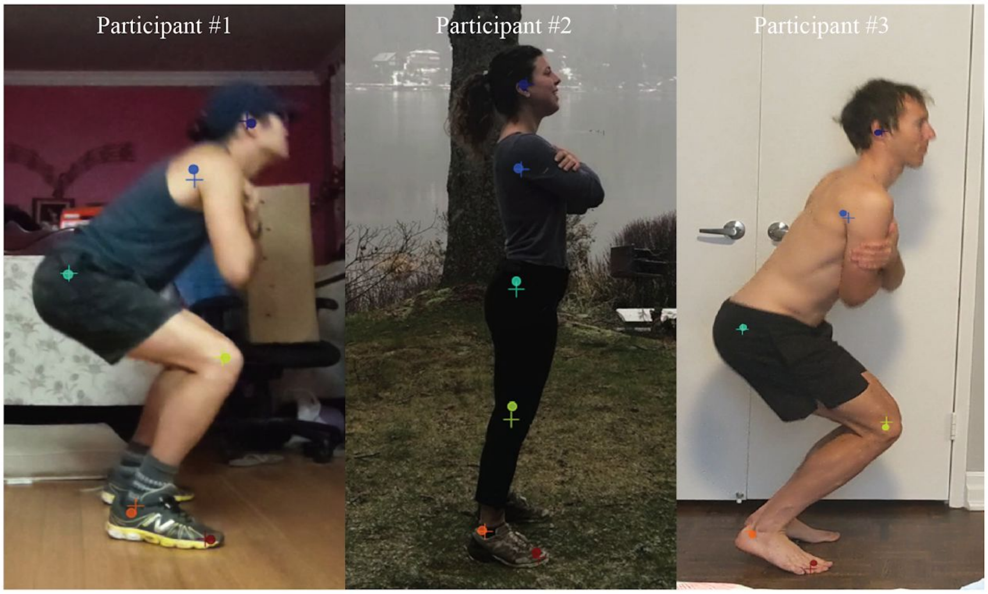

Figure 5: Three representative frames of data showing the manually labelled anatomical landmark (circles) and the neural network predicted landmark for three randomly selected participants.

With the small training, the trained network generalized well on novel participants

The small number of labelled frames used in training resulted in good generalization on novel participants. I evaluated the best performing neural network, Set 3, on its ability to generalize. I found that it well predicted the anatomical landmarks on the 3 novel participants. First, by visual inspection, the predictions well match what I would manually label (**Figure 6**). The predictions had high likelihood ratios on average > 0.95.

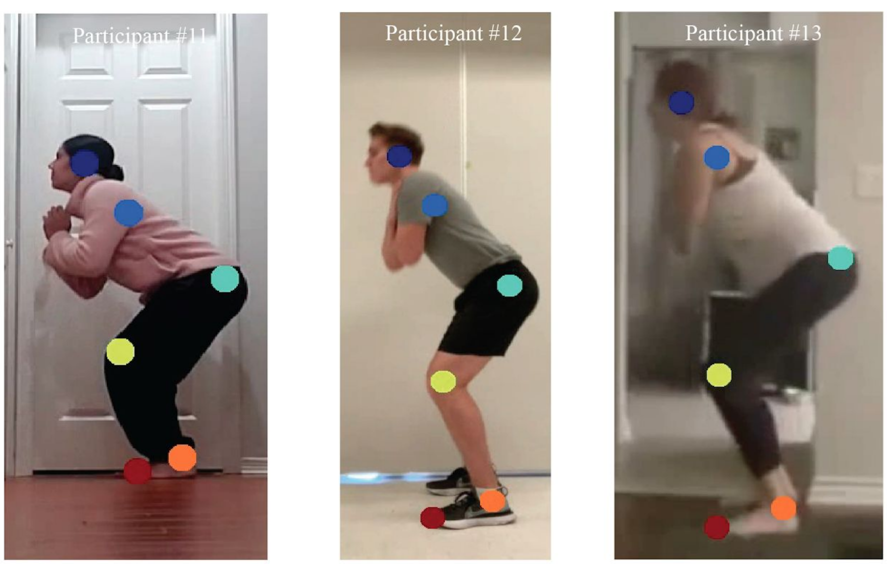

Figure 6: Representative frames of novel participants that the network was not trained on. Here the predicted anatomical landmarks (circles) are shown. I consider this performance quite good as it well matches what I would have predicted manually.

Sagittal plane joint kinematics

I found that the proposed approach resulted in typical sagittal plane joint kinematics normally found for vertical jumping tasks. I used the best performing neural network (Set 3) and the predicted anatomical landmarks to perform inverse dynamics analysis to evaluate the sagittal plane joint angles. The joint angles found well match that reported in the literature for vertical jumping [17]–[19] (**Figure 7**).

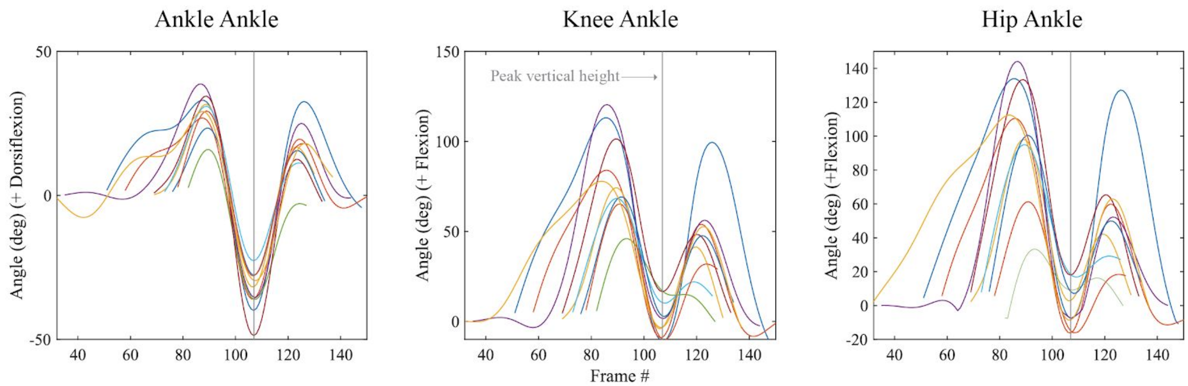

Figure 7: Sagittal plane joint angles determined for 10 participants (each coloured line represents a single participant). The joint angles are plotted to line up at the point where the participant has reached their peak vertical jump height as indicated by the vertical line.

Deep learning as a tool for biomechanics research

Identification of anatomical landmarks from video using deep learning allows for accessible biomechanical analysis of vertical jumping biomechanics. Here I provide an example of a jumping video containing the predicted anatomical landmarks overlayed onto the video, the skeleton of the person using the landmarks, and the predicted ankle, knee, and hip joints angles (**Figure 8, see supplementary data for video**). The purpose of this type of plot is to demonstrate the feasibility of using this type of approach for engineering teaching and learning. Although the movement studied here was vertical jumping, any movement can be studied in the classroom. Using DeepLabCut and inverse dynamic analysis is an opportunity for engineering students to get hands-on experience with relevant tasks while learning how to use vectors and dot products to determine angles. Not only does this create a motivation for the students, but it also creates engagement as they are actively involved in collecting, processing, and analyzing the data. A snapshot of 9 participants jumping is shown in **Figure 9**, demonstrating the power of using computer vision with deep learning to solve real-world biomechanics questions.

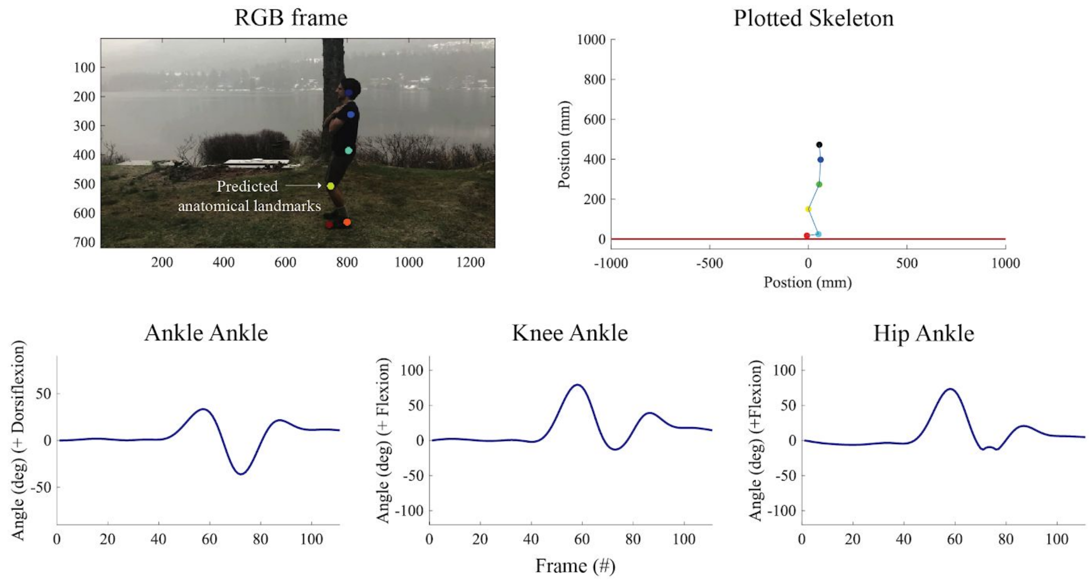

Figure 8: Example output showing the video with predicted anatomical landmarks, the plotted skeleton, and the sagittal plane ankle, knee, and hip joint angles.

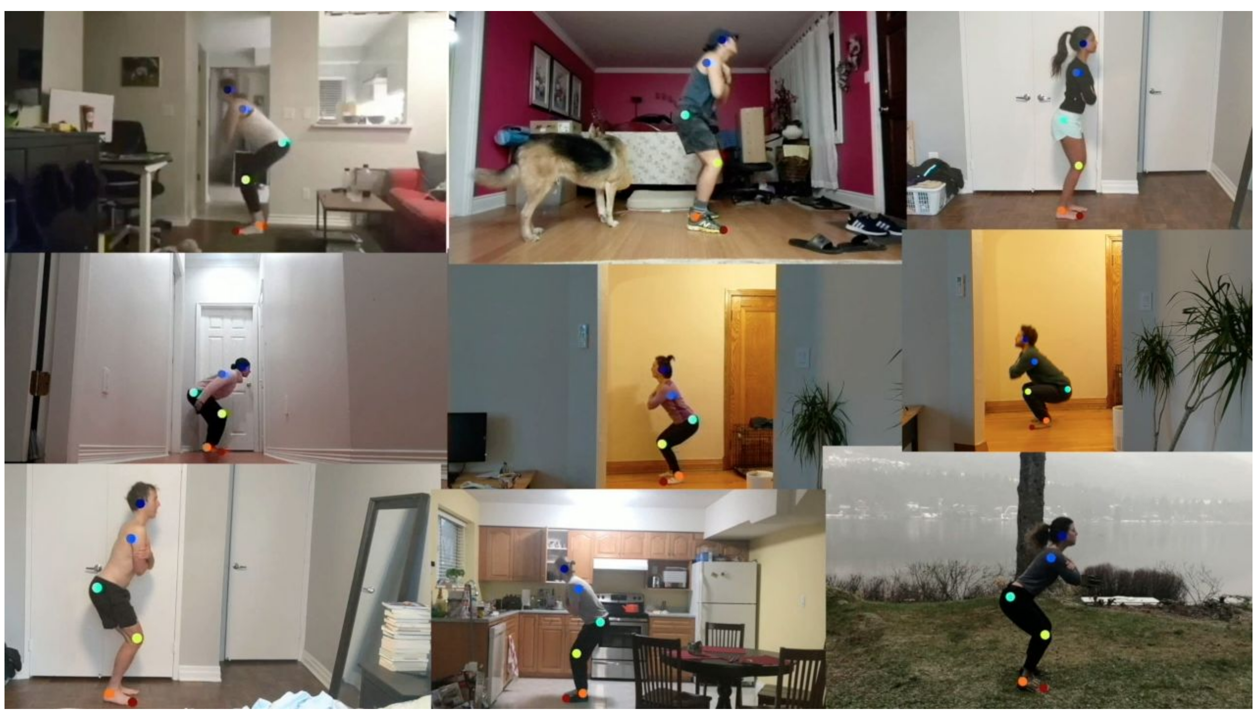

Figure 9: A snapshot of 9 participants and the neural network predictions of the 6 anatomical landmarks used in this research (video: https://www.youtube.com/watch?v=oqTJPbMpEDk).

.

Discussion

The results found here demonstrate the potential of using deep learning to study biomechanics and behaviour from video data. In this project, I was able to meet my two goals by achieving the four aims. First, I collected video data of people vertically jumping. With this data, I was able to successfully train a neural network to make predictions of 6 anatomical landmarks on the participant's bodies. The library DeepLabCut provided an easy to implement architecture for training a deep neural network using only a small sample of training data. I achieved my second goal by using the predicted landmarks to estimate the sagittal plane joint angles. I found that these predictions well matched joint angles reported in the literature. Thirdly, when using the trained neural network to make predictions on data it was not trained on I found that the network was well able to predict anatomical landmarks. The ability to generalize well when trained on only a small training set is notable and impressive. Finally, the results and the approach is taken here I believe make for a useful tool for teaching engineering students. I demonstrated this by providing an example interface for providing interactive outputs.

Deep learning using open source tools such as DeepLabCut is a rich opportunity for educational purposes in the engineering classroom. The project outcomes here demonstrate the potential for hands-on experiments for engineering students inside the physical classroom and in the developing virtual classroom. By using open-source tools, and compute provided for free through Google Colab, DeepLabCut provides a wide range of accessibility to all classrooms regardless of budget (Mathis et al. 2020; Google Colaboratory). The learning opportunities here extend beyond hands-on. By using neural networks for tracking video data, students can learn the general backbones of this approach while getting the experience of solving real-world problems. For engineers interested in biomechanics, a tool like DeepLabCut can be paired with pose estimation to teach and program inverse dynamics. Dynamics is the core of many engineering programs. While typically a dry topic, the opportunity to get experienced while learning first principles engineering approaches is exciting. With the growing field of wearable technology, the tools used here present an opportunity for pairing video data with wearable sensor data. The basic requirements of Python and a general understanding of computer programming make

There are several limitations in this research study that may be addressed in the future to further improve the results and the educational experience. First, all the work done here was done using only a single video from each participant at one time. By collecting 2 videos from different angles, the possibility of 3D anatomical landmark prediction and tracking is possible. DeepLabCut has integrated 3D toolboxes that were not explored here but provides the tools to make this happen (Nath et al. 2019). Furthermore, other sports or movements can easily be quantified (Fiker et al. 2020; Labuguen et al. 2019). Although I only looked at jumping mechanics here, any movement can be explored. The possibility of looking at cycling, running, swimming, and any motion that can be recorded provide limitless opportunities for learning human or animal behaviour. The ability to recruit and collect video data without the use of the lab also provides the opportunity for increasing the sample size without additional resources.

Work Cited

[1] W. J. Choi, J. M. Wakeling, and S. N. Robinovitch, "Kinematic analysis of video-captured falls experienced by older adults in long-term care," *J. Biomech.*, vol. 48, no. 6, pp. 911–920, Apr. 2015.

[2] S. N. Robinovitch, F. Feldman, Y. Yang, R. Schonnop, P. M. Leung, T. Sarraf, J. Sims-Gould, and M. Loughin, "Video capture of the circumstances of falls in elderly people residing in long-term care: an observational study," *Lancet*, vol. 381, no. 9860, pp. 47–54, Jan. 2013.

[3] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge,

"DeepLabCut: markerless pose estimation of user-defined body parts with deep learning," *Nat. Neurosci.*, vol. 21, no. 9, pp. 1281–1289, Sep. 2018.
[4] D. A. Winter, "Biomechanics and Motor Control of Human Movement." 2009.
[5] A. Toshev and C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks," *arXiv [cs.CV]*, 17-Dec-2013.
[6] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "DeeperCut: A Deeper, Stronger, and Faster Multi-person Pose Estimation Model," in *Computer Vision – ECCV 2016*, 2016, pp. 34–50.
[7] Z. Cao, G. Hidalgo Martinez, T. Simon, S.-E. Wei, and Y. A. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, Jul. 2019.
[8] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis, "Using DeepLabCut for 3D markerless pose estimation across species and behaviors," *Nat. Protoc.*, vol. 14, no. 7, pp. 2152–2176, Jul. 2019.
[9] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
[10] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, 2014, pp. 3686–3693.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv [cs.CV]*, 10-Dec-2015.
[12] Z. Lu, X. Jiang, and A. Kot, "Deep Coupled ResNet for Low-Resolution Face Recognition," *IEEE Signal Process. Lett.*, vol. 25, no. 4, pp. 526–530, Apr. 2018.
[13] A. Mathis, T. Biasi, Y. Mert, B. Rogers, M. Bethge, and M. W. Mathis, "Imagenet performance correlates with pose estimation robustness and generalization on out-of-domain data," in *International Conference on Machine Learning 2020 Workshop on Uncertainty and Robustness in Deep Learning*, 2020.
[14] A. Mathis, S. Schneider, J. Lauer, and M. W. Mathis, "A Primer on Motion Capture with Deep Learning: Principles, Pitfalls, and Perspectives," *Neuron*, vol. 108, no. 1, pp. 44–65, Oct. 2020.
[15] "Google Colaboratory." [Online]. Available: https://colab.research.google.com/notebooks/intro.ipynb. [Accessed: 04-Dec-2020].
[16] *DeepLabCut*. Github.
[17] K. B. Cheng, "The relationship between joint strength and standing vertical jump performance," *J. Appl. Biomech.*, vol. 24, no. 3, pp. 224–233, Aug. 2008.
[18] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine, "An optimal control model for maximum-height human jumping," *J. Biomech.*, vol. 23, no. 12, pp. 1185–1198, 1990.
[19] J. P. Van Zandwijk, M. F. Bobbert, M. Munneke, and P. Pas, "Control of maximal and submaximal vertical jumps:," *Med. Sci. Sports Exercise*, vol. 32, no. 2, p. 477, Feb. 2000.

Matlab Code for Inverse kinematics

```
close all; clear all; clc;
%%
if 1
```

```
    close all; clearvars -except p; clc;
if 1
    load('AllDeepLabCutData.mat')
end

SubNub = 10;

Sub = {'S1','S2','S3','S4','S5','S6','S7','S8','S9','S10','S11','S12','S13','S14','S15'};
SubjectName= Sub{1,SubNub}; % Change this to whatever you want to call it
% SubjectFolder = [ Sub{1,SubNub} '_' ,People{1,SubNub} ,'\'];
% drive = [Folder SubjectFolder];
AllData = [];


saveData = 1;
Subject = 10;
makevideo = 1;

%% Import Data

% Import Data

% import the whole csv file
% there are 4 dots that we care about
nameFile = [char(Sub{1,SubNub}) '.csv'];
DataImport = (readmatrix(nameFile))*-1;

if makevideo ==1
nameFile = [char(Sub{1,SubNub}) '.mp4'];
v1= VideoReader(nameFile);

myVideo = VideoWriter([nameFile '_outputFile']); %open video file
myVideo.FrameRate = 10;  %can adjust this, 5 - 10 works well for me
open(myVideo)

end


%% Calibrate the data so its at zero ref frame
Calibration = DataImport(1,17:18);
% Calibration = 0;
% Calibration = 0;
% Calibration =0;
temp.Head = DataImport(:,2:3)-Calibration; % x-y
temp.Body = DataImport(:,5:6)-Calibration; % x-y
temp.Hip = DataImport(:,8:9)-Calibration; % x-y
temp.Knee = DataImport(:,11:12)-Calibration; % x-y
temp.Ankle = DataImport(:,14:15)-Calibration; % x-y
temp.Toe = DataImport(:,17:18)-Calibration; % x-y

temp.LikeHood.Head = DataImport(:,4); % x-y
temp.LikeHood.Body = DataImport(:,7); % x-y
temp.LikeHood.Hip = DataImport(:,10); % x-y
temp.LikeHood.Knee = DataImport(:,13); % x-y
temp.LikeHood.Ankle = DataImport(:,16); % x-y
temp.LikeHood.Toe = DataImport(:,19); % x-y



%% Filter the markerdata using
[b,a] = butter(6, 6*2/150);
temp.Head= filtfilt(b, a,temp.Head);
temp.Body= filtfilt(b, a,temp.Body);
temp.Hip= filtfilt(b, a,temp.Hip);
temp.Knee= filtfilt(b, a,temp.Knee);
```

```
temp.Ankle= filtfilt(b, a,temp.Ankle);
temp.Toe= filtfilt(b, a,temp.Toe);

%% Determine the Joint Angles

% Ankle Angle
%Determine the segment vector from the ankle to the MP joint
%This is equal to the position vector of the MP minus the position
%vector of the ankle
rx_seg(:, 1) = temp.Ankle(:, 1) - temp.Toe(:, 1);
ry_seg(:, 1) = temp.Ankle(:, 2) - temp.Toe(:, 2);

%Determine the segment vector from the knee to the ankle
%This is equal to the position vector of the knee minus the position
%vector of the ankle
rx_seg(:, 2) = temp.Knee(:, 1) - temp.Ankle(:, 1);
ry_seg(:, 2) = temp.Knee(:, 2) - temp.Ankle(:, 2);

ry_unit(:, 1) = rx_seg(:, 1)./sqrt((rx_seg(:, 1).^2) + (ry_seg(:, 1).^2));
rz_unit(:, 1) = ry_seg(:, 1)./sqrt((rx_seg(:, 1).^2) + (ry_seg(:, 1).^2));

ry_unit(:, 2) = rx_seg(:, 2)./sqrt((rx_seg(:, 2).^2) + (ry_seg(:, 2).^2));
rz_unit(:, 2) = ry_seg(:, 2)./sqrt((rx_seg(:, 2).^2) + (ry_seg(:, 2).^2));

theta(:, 1) = acosd(ry_unit(:, 1));
theta(:, 2) = acosd(ry_unit(:, 2));

%Determine the joint angles
%Determine the dot product between rka and rma
ram_unit = [ry_unit(:, 1) rz_unit(:, 1)];
rka_unit = [ry_unit(:, 2) rz_unit(:, 2)];

rDot = dot(rka_unit, ram_unit, 2);

Angles.Ankle = acosd(rDot);

% Knee Angle
%Determine the segment vector from the ankle to the MP joint
%This is equal to the position vector of the MP minus the position
%vector of the ankle
rx_seg(:, 1) = temp.Knee(:, 1) - temp.Ankle(:, 1);
ry_seg(:, 1) = temp.Knee(:, 2) - temp.Ankle(:, 2);

%Determine the segment vector from the knee to the ankle
%This is equal to the position vector of the knee minus the position
%vector of the ankle
rx_seg(:, 2) = temp.Hip(:, 1) - temp.Knee(:, 1);
ry_seg(:, 2) = temp.Hip(:, 2) - temp.Knee(:, 2);

ry_unit(:, 1) = rx_seg(:, 1)./sqrt((rx_seg(:, 1).^2) + (ry_seg(:, 1).^2));
rz_unit(:, 1) = ry_seg(:, 1)./sqrt((rx_seg(:, 1).^2) + (ry_seg(:, 1).^2));

ry_unit(:, 2) = rx_seg(:, 2)./sqrt((rx_seg(:, 2).^2) + (ry_seg(:, 2).^2));
rz_unit(:, 2) = ry_seg(:, 2)./sqrt((rx_seg(:, 2).^2) + (ry_seg(:, 2).^2));

theta(:, 1) = acosd(ry_unit(:, 1));
theta(:, 2) = acosd(ry_unit(:, 2));

%Determine the joint angles
%Determine the dot product between rka and rma
ram_unit = [ry_unit(:, 1) rz_unit(:, 1)];
rka_unit = [ry_unit(:, 2) rz_unit(:, 2)];

rDot = dot(rka_unit, ram_unit, 2);
```

```
Angles.Knee = acosd(rDot);

% Hip angle
%Determine the segment vector from the ankle to the MP joint
%This is equal to the position vector of the MP minus the position
%vector of the ankle
rx_seg(:, 1) = temp.Hip(:, 1) - temp.Knee(:, 1);
ry_seg(:, 1) = temp.Hip(:, 2) - temp.Knee(:, 2);

%Determine the segment vector from the knee to the ankle
%This is equal to the position vector of the knee minus the position
%vector of the ankle
rx_seg(:, 2) = temp.Body(:, 1) - temp.Hip(:, 1);
ry_seg(:, 2) = temp.Body(:, 2) - temp.Hip(:, 2);

ry_unit(:, 1) = rx_seg(:, 1)./sqrt((rx_seg(:, 1).^2) + (ry_seg(:, 1).^2));
rz_unit(:, 1) = ry_seg(:, 1)./sqrt((rx_seg(:, 1).^2) + (ry_seg(:, 1).^2));

ry_unit(:, 2) = rx_seg(:, 2)./sqrt((rx_seg(:, 2).^2) + (ry_seg(:, 2).^2));
rz_unit(:, 2) = ry_seg(:, 2)./sqrt((rx_seg(:, 2).^2) + (ry_seg(:, 2).^2));

theta(:, 1) = acosd(ry_unit(:, 1));
theta(:, 2) = acosd(ry_unit(:, 2));

%Determine the joint angles
%Determine the dot product between rka and rma
ram_unit = [ry_unit(:, 1) rz_unit(:, 1)];
rka_unit = [ry_unit(:, 2) rz_unit(:, 2)];

rDot = dot(rka_unit, ram_unit, 2);

Angles.Hip = acosd(rDot);

%
% if SubNub ==1
% % Angles.Knee(55) = NaN
% Angles.Knee(55) = NaN
% Angles.Knee(56) = NaN
% Angles.Knee(57) = NaN
% Angles.Knee(58) = NaN
%
% for i = 1:size(Angles.Hip,2)
%     badi = find(isnan(Angles.Knee(:,i)));
%     goodi = find(~isnan(Angles.Knee(:,i)));
%     xnew = interp1(goodi, Angles.Knee(goodi,i), badi, 'nearest','extrap'); Angles.Knee(badi,i) = xnew;
% %     ynew = interp1(goodi, mr.y(goodi,i), badi, 'nearest','extrap'); mr.y(badi,i) = ynew;
% %     znew = interp1(goodi, mr.z(goodi,i), badi, 'nearest','extrap'); mr.z(badi,i) = znew;
% end
%
%
% Angles.Knee(55:58) = xnew;
%
% end
%% Anthropometrics
x1 = temp.Knee(1,1);
y1 = temp.Knee(1,2);

x2 = temp.Ankle(1,1);
y2 = temp.Ankle(1,2);

ShankLength = sqrt((y2-y1)^2+(x1-x2)^2);

x1 = temp.Ankle(1,1);
y1 = temp.Ankle(1,2);
```

```
x2 = temp.Toe(1,1);
y2 = temp.Toe(1,2);

FootLength =sqrt((y2-y1)^2+(x1-x2)^2);

x1 = temp.Hip(1,1);
y1 = temp.Hip(1,2);

x2 = temp.Knee(1,1);
y2 = temp.Knee(1,2);

ThighLength =sqrt((y2-y1)^2+(x1-x2)^2);

LegLength = ThighLength+ShankLength;

%% Lets plot and see the figure
figure('Renderer', 'painters', 'Position', [0 300 500 800]); hold on

floor = temp.Toe(1,:);

for i = 1:size(DataImport,1)

h = subplot(3,3,[1 1:3]);
  v1frame = v1.read(i);
  v1frame = flipdim(v1frame,2);
  limits = size(v1frame);
  image(v1frame, 'Parent',h)
    set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times

h1 = subplot(3,3,[4:6]); hold on

    h1.XLim = [ -1000 1000];
    h1.YLim = [ -50 1000];

    floorLim = h1.XLim;

        plot(temp.Head(i,1),temp.Head(i,2),'.k','MarkerSize',15)
        plot(temp.Body(i,1),temp.Body(i,2),'.b','MarkerSize',15)
        plot(temp.Hip(i,1),temp.Hip(i,2),'.g','MarkerSize',15)
        plot(temp.Knee(i,1),temp.Knee(i,2),'.y','MarkerSize',15)
        plot(temp.Ankle(i,1),temp.Ankle(i,2),'.c','MarkerSize',15)
        plot(temp.Toe(i,1),temp.Toe(i,2),'.r','MarkerSize',15)

        line ([temp.Head(i,1) temp.Body(i,1)],[temp.Head(i,2) temp.Body(i,2)] );
        line ([temp.Body(i,1) temp.Hip(i,1)],[temp.Body(i,2) temp.Hip(i,2)] );
        line ([temp.Hip(i,1) temp.Knee(i,1)],[temp.Hip(i,2) temp.Knee(i,2)] );
        line ([temp.Knee(i,1) temp.Ankle(i,1)],[temp.Knee(i,2) temp.Ankle(i,2)] );
        line ([temp.Ankle(i,1) temp.Toe(i,1)],[temp.Ankle(i,2) temp.Toe(i,2)] );

        line([floorLim(1) floorLim(2)],[floor(2) floor(2)],'color','r');

        h1.XLim = [ -1000 1000];
        h1.YLim = [ -50 1000];

        xlabel ('Postion (mm)')
        ylabel ('Postion (mm)')
        set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times


%           h1.XLim = [ 0 limits(1)];
%     h1.YLim = [ 0 limits(2)];
```

```
h2 = subplot(3,3,7); hold on;
     xlim([0 size(DataImport,1)])
     ylim([-90 90])
     dataplotAnkle(i) = Angles.Ankle(i)-(Angles.Ankle(1));
     time(i) = i;
     plot(time,dataplotAnkle','-')
     title('Ankle Angle')
     ylabel('Angle (deg) (+ Dorsiflexion)')

       set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times

h3 = subplot(3,3,8);hold on;
xlim([0 size(DataImport,1)])
ylim([-120 120])
dataplotKnee(i) = Angles.Knee(i)-(Angles.Knee(1));
plot(time,dataplotKnee','-')
title('Knee Angle')
ylabel('Angle (deg) (+ Flexion)')

xlabel('Frame (#)')
  set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times

h4 =subplot(3,3,9);hold on;
xlim([0 size(DataImport,1)])
ylim([-120 120])
dataplotHip(i) = Angles.Hip(i)-(Angles.Hip(1));
plot(time,dataplotHip','-')
title('Hip Angle')
ylabel('Angle (deg) (+Flexion)')
  set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times



drawnow

if makevideo ==1
  frame = getframe(gcf); %get frame
   writeVideo(myVideo, frame);
end

%     img =  frame2im(frame);
%     [img,cmap] = rgb2ind(img,256);
%     if i == 1
%         imwrite(img,cmap,'animation.gif','gif','LoopCount',Inf,'DelayTime',1);
%     else
%         imwrite(img,cmap,'animation.gif','gif','WriteMode','append','DelayTime',1);
%     end

if i ==size(DataImport,1)

else

cla(h1)
cla(h2)
cla(h3)
cla(h4)
```

```
end



end

if makevideo ==1
close(myVideo)
end



%% Save outout

AllDeepLabCutData.(char(SubjectName)).Angles(:,1) = Angles.Ankle;
AllDeepLabCutData.(char(SubjectName)).Angles(:,2) = Angles.Knee;
AllDeepLabCutData.(char(SubjectName)).Angles(:,3) = Angles.Hip;

AllDeepLabCutData.(char(SubjectName)).LikelyHood(:,1) = temp.LikeHood.Head;
AllDeepLabCutData.(char(SubjectName)).LikelyHood(:,2) = temp.LikeHood.Body;
AllDeepLabCutData.(char(SubjectName)).LikelyHood(:,3)= temp.LikeHood.Hip;
AllDeepLabCutData.(char(SubjectName)).LikelyHood(:,4) = temp.LikeHood.Knee;
AllDeepLabCutData.(char(SubjectName)).LikelyHood(:,5) = temp.LikeHood.Ankle;
AllDeepLabCutData.(char(SubjectName)).LikelyHood(:,6)= temp.LikeHood.Toe;

AllDeepLabCutData.(char(SubjectName)).Markers.x(:,1) = temp.Head(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.x(:,2) = temp.Body(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.x(:,3)= temp.Hip(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.x(:,4) = temp.Knee(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.x(:,5) = temp.Ankle(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.x(:,6)= temp.Toe(:,1);

AllDeepLabCutData.(char(SubjectName)).Markers.y(:,1) = temp.Head(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.y(:,2) = temp.Body(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.y(:,3)= temp.Hip(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.y(:,4) = temp.Knee(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.y(:,5) = temp.Ankle(:,1);
AllDeepLabCutData.(char(SubjectName)).Markers.y(:,6)= temp.Toe(:,1);

AllDeepLabCutData.(char(SubjectName)).Anthro.Foot = FootLength;
AllDeepLabCutData.(char(SubjectName)).Anthro.Shank = ShankLength;
AllDeepLabCutData.(char(SubjectName)).Anthro.Leg = LegLength;
AllDeepLabCutData.(char(SubjectName)).Anthro.Thigh = ThighLength;


save('AllDeepLabCutData.mat','AllDeepLabCutData')

end

%% Learning Stats
if 1

% LearningStats1 = (readmatrix('learning_stats_10000.csv'));
% LearningStats2 = (readmatrix('learning_stats_55000.csv'));
% LearningStats3 = (readmatrix('learning_stats_129000.csv'));


LearningStats1 = (readmatrix('learning_stats_55000.csv'));
LearningStats2 = (readmatrix('learning_stats_129000.csv'));
LearningStats3 = (readmatrix('learning_stats_220000.csv'));

figure();
subplot(1,3,1);hold on
plot(LearningStats1(:,1), LearningStats1(:,2),'-r')
plot(LearningStats2(:,1), LearningStats2(:,2),'-b')
```

```
plot(LearningStats3(:,1), LearningStats3(:,2),'-g')
legend('Set1','Set2','Set3')
xlabel('Iteration #')
ylabel('Loss')
ylim([0 0.07])
xlim([100 10e4])
        set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times

subplot(1,3,2); hold on

x = [1 2 3];
vals = [LearningStats1(10,2)  LearningStats1(500,2)  LearningStats1(end,2);   LearningStats2(2,2)  LearningStats2(500,2) LearningStats2(end,2) ;...
    LearningStats3(1,2)  LearningStats3(500,2) LearningStats3(end,2)];
b = bar(x,vals);
ylim([0 0.07])
        set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times
ylabel('Loss')


% LearningStats1 = (readmatrix('DLC_10000-results.csv'));
LearningStats1 = (readmatrix('DLC_57000-results.csv'));
LearningStats2 = (readmatrix('DLC_129000-results.csv'));
LearningStats3 = (readmatrix('DLC_220000-results.csv'));

Test(1,1) = LearningStats1(5);
Test(1,2) = LearningStats2(5);
Test(1,3) = LearningStats3(5);

Train(1,1) = LearningStats1(6);
Train(1,2) = LearningStats2(6);
Train(1,3) = LearningStats3(6);

subplot(1,3,3); hold on

x = [1 2 3];
vals = [Test; Train];
b = bar(x,vals);
% ylim([0 0.07])
        set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times
ylabel('Pixel Error')

end

%% Plotting all data analysis
if 1
load('AllDeepLabCutData.mat')
Sub = {'S1','S2','S3','S4','S5','S6','S7','S8','S9','S10'};

figure(); hold on
matrixOfPlot.Ankle = zeros(300,10);
matrixOfPlot.Knee = zeros(300,10);
matrixOfPlot.Hip = zeros(300,10);
matrixOfPlot.LikelyHood.Body= zeros(300,10);
matrixOfPlot.LikelyHood.Hip= zeros(300,10);
matrixOfPlot.LikelyHood.Knee= zeros(300,10);
matrixOfPlot.LikelyHood.Ankle= zeros(300,10);
matrixOfPlot.LikelyHood.Toe= zeros(300,10);
for i = 1:10
```

```
    subplot(2,6,i)
    theta_ankle= AllDeepLabCutData.(char(Sub(i))).Angles(:,1);
    [high,ind(i)] = min(theta_ankle);

    temp = size(theta_ankle, 1);
    time = (temp/100):(100/temp):100;
    plot(time, theta_ankle(1:length(time))')

    sizes(i) = temp;


    clear time temp theta_ankle


%     plot(t,x-x(1),'-')


end

for i =1:10
    theta_ankle= AllDeepLabCutData.(char(Sub(i))).Angles(:,1)-AllDeepLabCutData.(char(Sub(i))).Angles(1,1);
    theta_knee= AllDeepLabCutData.(char(Sub(i))).Angles(:,2)-AllDeepLabCutData.(char(Sub(i))).Angles(1,2);
    theta_hip= AllDeepLabCutData.(char(Sub(i))).Angles(:,3)-AllDeepLabCutData.(char(Sub(i))).Angles(1,3);


%       temp = size(theta_ankle, 1);
    if i ==1
    matrixOfPlot.Ankle(50:(length(theta_ankle)+49),i) = theta_ankle;
    matrixOfPlot.Knee(50:(length(theta_knee)+49),i) = theta_knee;
    matrixOfPlot.Hip(50:(length(theta_hip)+49),i) = theta_hip;
    matrixOfPlot.LikelyHood.Body(50:(length(theta_hip)+49),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,2);
    matrixOfPlot.LikelyHood.Hip(50:(length(theta_hip)+49),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,3);
    matrixOfPlot.LikelyHood.Knee(50:(length(theta_hip)+49),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,4);
    matrixOfPlot.LikelyHood.Ankle(50:(length(theta_hip)+49),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,5);
    matrixOfPlot.LikelyHood.Toe(50:(length(theta_hip)+49),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,6);

    else
    A = ind(1)-ind(i) +50;
    matrixOfPlot.Ankle(A:(length(theta_ankle)+A-1),i) = theta_ankle;
    matrixOfPlot.Knee(A:(length(theta_knee)+A-1),i) = theta_knee;
    matrixOfPlot.Hip(A:(length(theta_hip)+A-1),i) = theta_hip;
    matrixOfPlot.LikelyHood.Body(A:(length(theta_hip)+A-1),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,2);
    matrixOfPlot.LikelyHood.Hip(A:(length(theta_hip)+A-1),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,3);
    matrixOfPlot.LikelyHood.Knee(A:(length(theta_hip)+A-1),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,4);
    matrixOfPlot.LikelyHood.Ankle(A:(length(theta_hip)+A-1),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,5);
    matrixOfPlot.LikelyHood.Toe(A:(length(theta_hip)+A-1),i)= AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,6);

    end

      clear time temp theta_ankle
end

  matrixOfPlot.Ankle(matrixOfPlot.Ankle==0)= NaN;
   matrixOfPlot.Knee(matrixOfPlot.Knee==0)= NaN;
    matrixOfPlot.Hip(matrixOfPlot.Hip==0)= NaN;
    matrixOfPlot.LikelyHood.Body(matrixOfPlot.LikelyHood.Body ==0) = NaN;
    matrixOfPlot.LikelyHood.Hip(matrixOfPlot.LikelyHood.Hip ==0) = NaN;
    matrixOfPlot.LikelyHood.Knee(matrixOfPlot.LikelyHood.Knee ==0) = NaN;
    matrixOfPlot.LikelyHood.Ankle(matrixOfPlot.LikelyHood.Ankle ==0) = NaN;
     matrixOfPlot.LikelyHood.Toe(matrixOfPlot.LikelyHood.Toe ==0) = NaN;

    figure();
    subplot(1,3,1)
    plot(matrixOfPlot.Ankle)
            set(gca,'FontSize',16)
```

```
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times
      ylabel('Angle (deg) (+ Dorsiflexion)')

      line([ 107 107], [-50 50])
      ylim([-50 50])


      subplot(1,3,2)
   plot(matrixOfPlot.Knee)
            set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times
      ylabel('Angle (deg) (+ Flexion)')
      xlabel('Frame #')

      line([ 107 107], [-10 150])
      ylim([-10 150])

      subplot(1,3,3)
   plot(matrixOfPlot.Hip)
            set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times
ylabel('Angle (deg) (+Flexion)')
      line([ 107 107], [-20 150])
      ylim([-20 150])


end


%% Likely hood

if 1
load('AllDeepLabCutData.mat')
Sub = {'S1','S2','S3','S4','S5','S6','S7','S8','S9','S10'};

figure(); hold on

subplot(1,5,1)
histogram((matrixOfPlot.LikelyHood.Body*-1))
title('Body Marker')
ylim([0 700])
xlim([0.5 1.2])

subplot(1,5,2)
y1= histogram(matrixOfPlot.LikelyHood.Hip*-1)
ylim([0 700])

subplot(1,5,3)
y2= histogram(matrixOfPlot.LikelyHood.Knee*-1)
ylim([0 700])

subplot(1,5,4)
y3= histogram(matrixOfPlot.LikelyHood.Ankle*-1)
ylim([0 700])

subplot(1,5,5)
histogram(matrixOfPlot.LikelyHood.Toe*-1)
ylim([0 700])


%%
figure(); hold on
```

```
Sub = {'S1','S2','S3','S4','S5','S6','S7','S8','S9','S10'};
for i =1:10
    len = size(AllDeepLabCutData.(char(Sub(i))).LikelyHood,1);
x = [2*ones(1,len); 4*ones(1,len);6*ones(1,len);8*ones(1,len);10*ones(1,len);12*ones(1,len)];



y = [AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,1),AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,2), ...
    AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,3),AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,4),...
    AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,5),AllDeepLabCutData.(char(Sub(i))).LikelyHood(:,6),];

swarmchart(x(1,:),(y(:,1)*-1))
swarmchart(x(2,:),(y(:,2)*-1))
swarmchart(x(3,:),(y(:,3)*-1))
swarmchart(x(4,:),(y(:,4)*-1))
swarmchart(x(5,:),(y(:,5)*-1))
swarmchart(x(6,:),(y(:,6)*-1))

clear x y
end

ylim([0 1.01])

                set(gca,'FontSize',16)
set(gcf,'renderer','Painters')
set(gca,'fontname','times')  % Set it to times
        ylabel('Probablity')
view([-90 -90])




end
```

NOTES

Notes

Go to terminal and type : conda activate DLC-CPU

How to run the GUI
Running a virtual environment

conda activate tf

Pythonw

import deeplabcut

deeplabcut.launch_dlc()

https://colab.research.google.com/drive/1Nppc-58ZlyULKiX2KszIP0pv1FIG88Vt#scrollTo=Y_LZiS_0oEJl

I ran 10,000 iterations it took about 90 min and didnt work too well

pip install tensorflow==1.15

#
/usr/local/lib/python3.6/dist-packages/deeplabcutcore/pose_estimation_tensorflow/models/pretrained/resnet_v1_50.ckpt

https://colab.research.google.com/drive/1qUmMGfXii6VFWuIfiCITDvxziqbcnlS8#scrollTo=6aDF7Q7KoEKE