DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU

HARVARD LIBRARY
Office for Scholarly Communication

Live Perception and Real Time Motion Prediction with Deep Neural Networks and Machine Learning

Citation
Zielinski, Edward. 2021. Live Perception and Real Time Motion Prediction with Deep Neural Networks and Machine Learning. Master's thesis, Harvard University Division of Continuing Education.

Permanent link
https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37370061

Terms of Use
This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

Live Perception and Real Time Motion Prediction with Deep Neural Networks and Machine Learning

Edward Zielinski

A Thesis in the Field of Software Engineering

For the Degree of Master of Liberal Arts in Extension Studies

Harvard University

November 2021

Copyright 2021 Edward Zielinski

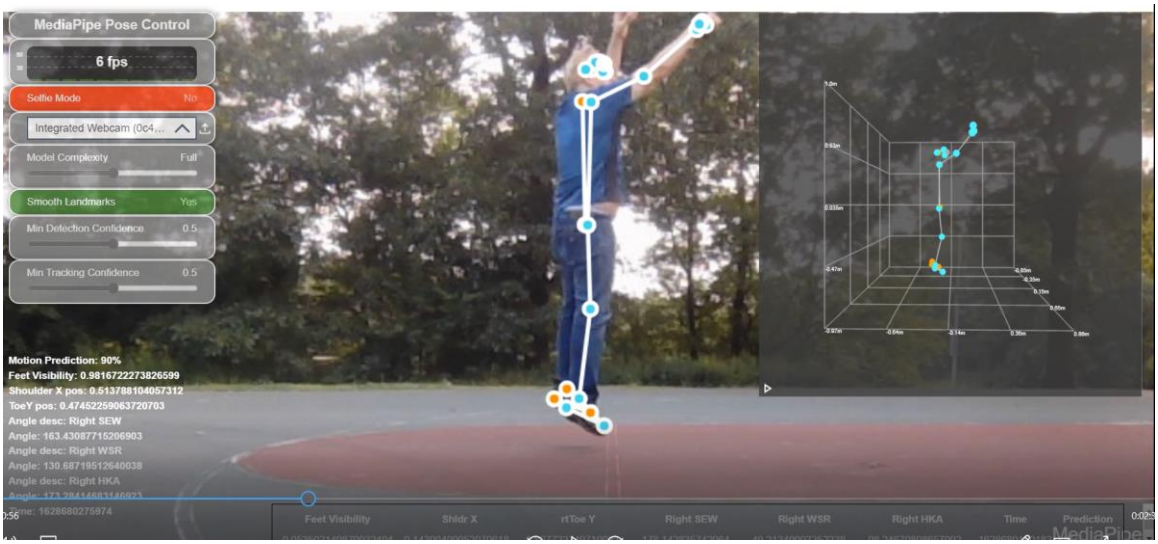
Abstract

The research in this project explores the intersection of human computer interaction (HCI) and deep neural networks. Advances in real-time output has reduced latency making webcam skeletal model output useful for fine motor skill motion research. The newer Live Perception model no longer relies on distant servers resulting in reduction of both latency and privacy issues. Here we take advantage of the advances and develop an interface with low latency and increased privacy to make predictions and inferences entirely with local processing. The interface customizes JavaScript on the client browser to use MediaPipe Pose, TensorFlow.js and Python's Keras. We call the new interface the Foul Shot Training Mirror. The live perception application provides a blueprint to create motion predictions from deep computer vision models by customizing the real time output.

Using this interface, researchers can create time series analysis with the real-time data. This advances HCI research by analyzing how a tight feedback loop can improve the fine motor skills involved in shooting a basketball. Our methods train sequenced motion data from real-time vision models to optimize Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Our research shows this novel approach to training the motion data is successful in training a Gated Recurrent Unit (GRU). In this new approach, we successfully implement a prototype and use computer vision data from the skeletal data points and angular velocities to predict motion from the deep learning simulation data.

The interface will scale to other applications by using real-time results in a more private and efficient manner. The predictions provide immediate feedback, allowing for immediate forward and backward chain learning. This style of learning aids in improving fine motor skills, which can be used in other research, such as to improve motor skills of people with injuries or disabilities, or to monitor and maintain proper motor skills as people age.

Frontispiece



By instantly projecting near real-time image and motion statistics in front of the foul shooter, they acquire a visual representation of how their mind is affecting their body's motor skills. From this perception, they immediately integrate the acquired knowledge and adjust their motor skills to change shooting motions to become a more consistent foul shooter (Appendix 3. **Projection Diagram**).

Author's Biographical Sketch

Edward Zielinski is a recent graduate of Harvard University's Master's Degree program in the Field of Software Engineering of Liberal Arts in Extension Studies and recipient of a Graduate Level Certificate in Data Science. Prior to receiving his Master's Degree, he studied Business Administration, Applied Mathematics and Economics at the University of New Hampshire, Durham and received a Bachelor's Degree in Business Administration with Minors in both Applied Mathematics and Economics. Other educational activities include a Calculus Grader and the Managing Editor for the Journal of Educational Computing Research. Professionally, he enjoys helping others understand difficult abstract concepts and keeping up with the latest in data science and machine learning fields. Industry experience includes over fifteen years of software development in a variety of industries to include Government, Education, Finance, Insurance, Energy, Banking, Manufacturing, Human Resources and Marketing. He envisions many untapped opportunities to advance machine learning and data science for these and many more industries.

Dedication

In my youth, I took to athletics, and to encourage my reading and scholarly activities my Mom and Dad would bring me to the library to pick out books. The books I read were 'How to' books. How to swing a baseball bat, How to play Hockey, How to Fish - you get the idea. Fast forward to today, and to encourage youngsters to read, learn and be active, I envision a new type of 'How to book', one that incorporates the latest in machine learning and deep neural networks. This new type of book with interactive application will teach a person many new skills to help develop motor skills and connect with their mind and body. While developing a healthy connection, they also get an appreciation of how modern technology and mathematics inter-relate with sports. For this reason, I dedicate this research to my Mom and Dad for their inspiration, and hope youngsters, and the young-at-heart, will be inspired and learn 'How to' perform the many different skills offered by this new 'How to' series. The first book in development is How to Shoot a Basketball Free Throw Shot. I would be remiss for not mentioning the continued support from my wife Maggie Zielinski. She is very understanding and despite the amount of time required to finish this research, always offered encouragement, so I sincerely thank her.

Acknowledgments

I would like to express my sincere gratitude to my Research Advisor, Dr. Hongming Wang, who gave excellent guidance and advice during the entire thesis process. Dr. Wang's guidance and structured one-on-one advising helped to structure a strong foundation for my thesis. Her guidance kept me on track and focused on the latest technology in the field of deep neural networks. Her input transcended my many thoughts and ideas into a concrete and grounded path. To my Thesis Director, Dr. Bakhtiar Mikhak, whose experience and wisdom is seemingly perpetual, I sincerely thank. His exceptional guidance from beginning to end is comforting and offers assurance in the difficult task of staying on tempo in the thesis process. Dr. Mikhak's style is informative and enlightening to say the least. Through the many twists and turns my thesis took, his guidance was always true and valuable. Finally, thank you to Harvard University's Extension School for making these extremely talented and dedicated people available to me during this process. This would not be possible without the valuable service and structure you offer.

Table of Contents

Abstract iii

Frontispiece v

Author's Biographical Sketch vi

Dedication vii

Acknowledgments viii

Chapter I Background and Significance 1

1.1 Historical Perspective – Deep Neural Networks 2

1.2 Current Research 5

1.3 HCI and Low Latency 9

Chapter II HCI and Visual Interface Design 10

2.1 Machine Learning Design Methods 10

2.2 The Foul Shot Mirror – Interface Design 14

Chapter III Data Collection, Processing and Training 16

3.1 Video Streaming Data 16

3.2 Data for Motion Monitoring 18

Chapter IV Skeletal Models and Experiments 22

4.1 DNN Skeletal Architectures 25

4.2 BlazePose architecture 28

4.2.1 BlazePose Skeletal Diagram 29

4.3 Extracting Angular Velocity from MediaPipe Pose 29
4.3.1 Data Output Diagram 31
Chapter V Baseline Models 33
5.1 Three Key Positions 34
5.2 Green Angle 35
5.3 Orange Angle 35
5.4 Green and Orange Angle changing over time 36
Chapter VI CNN and RNN Models 37
Chapter VII Results and Discussion 39
7.1 Choosing a Model 39
7.1 Usability 44
7.2 Scaling to Other Basketball Shots 44
Chapter VIII Conclusion 46
Appendix 1. Code Highlights 48
Video Scraping Code 48
Greyscale 48
Foul Shot Action Classifier Code 49
PoseNet 50
BlazePose customization 51
Angular Calculation 52
Classical Style Programming 52
TensorFlow Model Predictor 53
Appendix 2. Application and Data 54

Appendix 3. Projection Diagram 55

Appendix 4. Definition of Terms 57

Appendix 5. Initial Video Trials 60

Appendix 6. Common Errors in Free Throw Shooting 61

Appendix 7. Summary of Model Layers 62

Vectorize Data 62

RNN 62

LSTM 63

GRU 63

Appendix 8. Set-Up Instructions 65

Figures 66

Figure 1. PoseNet Skeletal Diagram 66

Figure 2. BlazePose Skeletal Diagram 67

Chapter I

Background and Significance

In 1960, Dr. Joseph Carl Robert Licklider, a computer scientist and psychologist wrote "Man-Computer Symbiosis" (Licklider, 1960). In his paper, he outlined the prerequisites for the relationship between man and computer as a symbiosis. He stated, "The information-processing equipment, for its part, will convert hypotheses into testable models." This interesting statement made sixty years ago, are the beginnings of the models for data science, and the foresight of deep neural networks. Today's digital media technology in video, audio, and text combined with Deep Neural Network algorithms are furthering the human-computer symbiosis that Licklider laid out in his 1960 analysis. Licklider made important observations on how the simple clerical and mechanical tasks to complete research will soon be computer-driven tasks. Our current research will show how these tasks are disappearing due to modern digital machines performing these tasks in near real time.

Today, machines use deep neural networks and analysis by utilizing the many digital sensors in vision, audio and touch. Significant advances allow the use of GPU speed and computational capability by using Web Graphics Libraries (WebGL) and Web Assembly (WASM) on client machines. This speed and computational ability on the client-side allow communication of the APIs with JavaScript and use the browser to create advanced simulations with the aid of deep neural networks. These continued advancements show the drive toward human-computer symbiosis as a driving research

force. Licklider anticipated that the computer would take over these menial repetitive tasks, and today we see the significance of Deep Neural Networks as a representation of the 'menial repetitive tasks' in which Licklider spoke about. Before getting into the design and calibrations of our digital Foul Shot Training Mirror, it is helpful to understand the history and development of the Deep Neural Network algorithms.

1.1 Historical Perspective – Deep Neural Networks

In the very beginning, Neural Networks (NN) were simple and essentially variants of linear regression. In 1962, there are references to a cat's visual cortex, which inspired deep neural network architectures. In 1971, Ivakhnenko discussed the Group Method for Data Handling (GMDH), with eight layers. The Neocognitron, by Fukushima in 1979 is the first to speak of neurophysiological insights of Convolutional Neural Networks (CNN). Back Propagation (BP) received significant recognition in 1986 from Rumelhart by experimentally demonstrating the use of hidden layers. It is important to realize the underpinnings of back propagation come from gradient descent work that started from Leibniz (1676), L'Hopital (1696), and Jacobian Matrix (1845) working in multivariate spaces. During the 1990s and 2000s, many research issues were dealing with the long-time lag problem of decaying and exploding gradients. Solutions examining the gradient issue overcame the issue with more computer power with GPUs, and a new model called Long/Short-Term Memory (LSTM) which deals with the time steps to resolve the vanishing gradient issue (Schmidhuber, 2015). The deep neural networks of today use deeper hierarchical levels of learning; they are Recurrent Neural Network (RNN), Feed Forward Neural Networks (FFNN), Long/Short-term Memory neural networks (LSTM),

and Convolutional Neural Networks (CNN). After 2010, RNN, CNN and LSTM won many awards to make these models among the top focus of research. In addition, in 2014 the Gated Recurrent Unit (GRU) was developed and is similar to LSTM, but is somewhat cheaper to run and in some cases more accurate than LSTM (Junyoung Chung et al, December 2014).

The DNNs for computer vision underwent several revisions of Convolutional Neural Networks (CNNs). Beginning in 1998 the LeNet-5 architecture for a CNN proposed five weight layers, three convolutional layers and two fully connected layers and had 61,706 parameters (Elgendy, 2020). In LeNet-5, a major change was to remove fully connected layers and instead use locally connected layers "where each plane is a feature map" (Lecun et al, 1998). CNN's extract meaningful features that separate an object from other images in the training set, and stack them in an array of features (Elgendy, 2020). In 1998, this was novel, but the LeNet-5 only could classify grey scale images and could only classify ten classes. In 2012, along came the AlexNet model that could classify 1,000 different classes, has about 60,000 parameters and 650,000 neurons and it had a larger learning capacity to understand more complex features (Elgendy, 2020), (Krizhevsky et al, 2017). In 2014, Visual Geometry group at Oxford University created VGG16 with even deeper layers (Elgendy, 2020). Next, in 2015, the Microsoft Research team solved the problem of the vanishing gradient with ResNet (K. He et al, 2016). Next, Google creates MobileNet, which significantly reduces the number of parameters and balances accuracy while restricting resources so it can work on many digital media devices (Pujara, 2020).

The current architecture of MobileNet allows for comparable results to ResNet and allows for a quicker response that allows for more real-time feedback (Pujara, 2020). So we can see the jumps in technology from 1998 to 2012, then 2015 with ResNet and in the following years Google developed MobileNet versions 1, 2, and 3 (Kulkarni et al, 2021). In May 2021, Google announced a next generation pose detection with MoveNet, which has two variants, known as Lightning and Thunder. "Lightning is intended for latency-critical applications, while Thunder is intended for applications that require high accuracy. Both models run faster (30+ FPS) on most modern desktops, laptops, and phones, which proves crucial for live fitness, sports, and health applications" (Votel Ronny, 2021). BlazePose is a current architecture released with MediaPipe in December 2019 "that produces 33 body key points for a single person and runs at over 30 frames per second on a Pixel 2 phone" (Valentin Bazarevsky et al, 2020). The current release used for this application is MediaPipe version 1.1624666670 released in June of 2021 (NPM, 2021). Google classifies the MediaPipe Pose model as Live Perception, which is any kind of machine learning that happens in the viewfinder. This means it happens on the device, in real time, and with low latency. All computation with Live Perception happen on the device with no connection to the internet, making the solution very privacy conscious (Brunham, Live Perception for Mobile and Web, 2020).

1.2 Current Research

The focus of this research is to determine the best Deep Neural Network algorithms to examine human motion and teach a human to enhance their fine motor skills. The motor skills to learn are the proper motions to become a better and more consistent basketball free-throw shooter. The main digital machine device to collect data for this study is the webcam, which falls under the research category of deep learning for vision systems. The current research findings for computer vision reveal that researchers primarily focus on certain aspects of a CNN model. For instance, a CNN research study may focus solely on updating the accuracy of a skeletal model (See Figures, Figure 1 and Figure 2), or classifying certain objects at a high confidence level. In all, current research in computer vision weighs more heavily toward object detection of a still image or object detection in a video stream. A secondary focus for my research is to examine the output of CNN's model data to predict different types of motion and determine if these motions result in a successful foul shot. For this secondary focus, we must examine a second type of DNN, the Recurrent Neural Network, or RNN.

"The RNN processes sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far." (Chollet, Deep Learning with Python, 2018, p. 196) Most research studies with RNN are for text analysis, where the focus is using previous text information to summarize a document, or to predict the next word (Yang et al, 2021). This study, however, will use RNN models to bring a memory state to the output of the computer vision data generated by the CNN model. The customization of the RNN model to analyze the motion data becomes the focus in Chapter VI of our research. Accelerometers are another possible interest for

research, because digital media devices like smartphones use an accelerometer and play a significant role in recognizing an individual's physical activity. There are many studies for classification of different activities, and some research even includes basketball shots as an activity (Lu et al, 2017). However, most accelerometer studies classify actions or objects and do not predict future positions based on past results, and for this study, deep learning for vision systems will remain the primary focus, with RNN's as the secondary data feed to our motion prediction model. Therefore, the gap in the research for motion studies with CNNs, unlike text studies, is in predicting a future result by using previous motions. We will use the concept of predicting the next letter result of a word and implement an RNN by analyzing the motion data output from our webcam CNN model to predict the accuracy of a basketball foul shot. The key experiment to help understand this gap involves using the data output from the skeletal Pose model and training the data with various types of RNN deep neural network models.

To summarize, current research of CNNs classify objects or actions to identify activities more accurately, but there is not much research predicting outcomes of motions, or actions. In other words, there is minimal research on Deep Neural Networks to optimize human motion to interact in a symbiotic way and teach a human. This research and application intend to help fill this gap with Google's ecosystem. Current technology with Google's MediaPipe and TensorFlow allow the designer to use fully developed pre-trained models to recognize key points on a human image. To develop the model and web application in this paper we use models from PoseNet in TensorFlow and Pose in MediaPipe (Oved et al, 2018), (Google, MediaPipe Pose, 2020). These pre-trained skeletal models use DNNs to classify the key points to create the superimposed skeletal

model of a person from a video streaming image. Using the output of these key-points from the streaming image, we feed the output into our RNN model. My development vision is to use machine-learning algorithms to customize the feedback to predict and identify key motions that will refine the motor skills to shoot a foul shot. The output will teach the user to become more cognizant of their fine motor skills and enable them to send the basketball from the foul line into the basket in a more consistent fashion. The Google ecosystem in machine learning provides the ability to customize the given pre-trained models, and allows the deep learning engineer to add customizations when a specific domain requires customization. With the ability to customize the application, we can now begin to explore the research gap as to whether a video device can learn the motions of a human, and then monitor or teach the necessary motor skills to improve the motion. Comparing this to traditional ways to learn to shoot a basketball is now making it apparent that the machine can notice the nuances of human motion to teach fine motor skills. The traditional way to learn is from a trained professional gathering data through visual observations and offering advice from years of experience and intuitive judgement. The focal point or guiding theory comes from words inside of Liklider's original paper. He states a good measure of success will show 'how' our machine learning DNNs can come up with a "suggested course of action that agrees with the man's intuitive judgment" (Licklider, 1960). In other words, we want to show our deep learning model is a substitute, or at least an enhancement, to the traditional method of learning to develop motor skills in performing certain tasks.

Our research goes beyond recognizing 'only' human activities and focuses on design methods to detect motion with the latest models from the Google Deep Neural

Network ecosystem of TensorFlow.js and MediaPipe. It also examines the human learning process of training the body to learn and optimize motor skills. Increasing the likelihood of scoring a foul shot is one result, but the process of increasing motor skills with deep neural networks is a result that adds even more value to this research. Showing we can increase an individual's motor skills to obtain a target goal, scales for many more domain applications, thus adding value to the current research. Human learning is a vast field in itself, but in the domain of a basketball foul shot, effects on learning show potential for research (Mark R. Wilson et al, 2009). Our study of the basketball foul shot is in a practice situation to deliver a baseline metric for measuring achievement of fine motor skills. This baseline is the starting point for more advanced studies in the developing field of machine learning as an application to enhance fine motor skills.

Our research develops the learning-feedback-loop by using the latest DNN models and near real time browser manipulation with computer vision overlaying a skeletal model onto a person shooting a basketball foul shot. The feedback loop emulates human cognition and amplifies the learning process. This type of forward chaining, exploits feature maps from our existing DNN models, to help "human beings acquire, represent, and integrate knowledge" (Campero et al, 2018). The justification of our research is to add value and increase methods to use the real time information to make a human more cognizant and aware, both somatically and at a subconscious proprioception type of level. The result will mesh the latest technology into a type of personal robot assistant to show advances toward a human-machine symbiosis. The goal is to showcase these advances of machine learning, deep learning and live perception toward the human-machine symbiosis discussed in Licklider' s paper.

The broader implications and justifications for this study pertain to the development and design of personal robot assistants for sports analytics, recoveries from traumatic injuries, geriatrics, and many other domains where fine motor skills are involved.

1.3 HCI and Low Latency

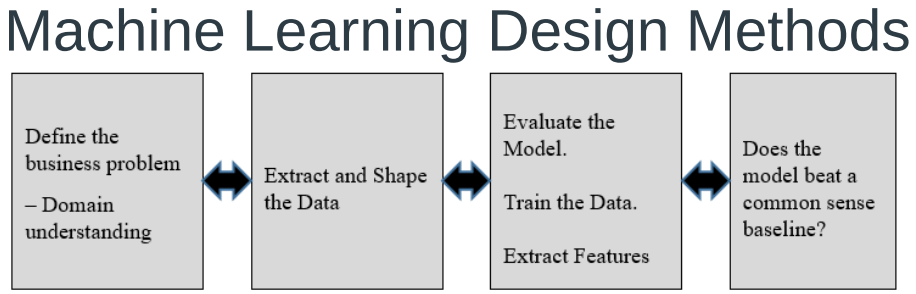
Twenty years from Licklider's explanation of symbiosis for man and computer, the personal computer movement is born, and helped to rename his concept to Human Computer Interaction (HCI). The personal computer inspired HCI research to make interaction easier for the non-computer savvy users. The next advent of human computer symbiosis is with the advances in Live Perception. The reduction of latency allows more code customization to add predictor functions from TensorFlow.js to predict motion. Predicting motion from skeletal models opens up research for motion prediction. The following chapter shows how human vision symbiotically connects with a real-time deep learning visual interface to represent the human cognitive experience. The interface we develop is part of the active research for the connectedness for envisioning body movements from the representations of CNN and RNN models. Metaphorically, computer vision from the webcam connects with human vision to enhance motor skills in real-time. This is the motivation to design a useful interface to interact with real-time data using CNN and RNN to create the Foul Shot Training Mirror. Human Computer Interaction combined with low latency computer vision opens a path for real cognitive experiences to enhance fine motor skills.

Chapter II

HCI and Visual Interface Design

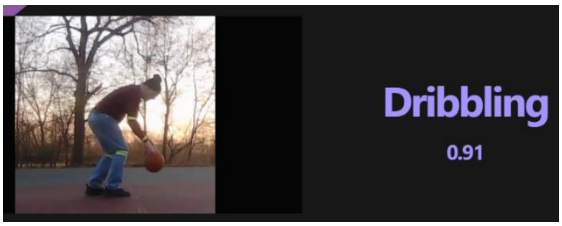
2.1 Machine Learning Design Methods

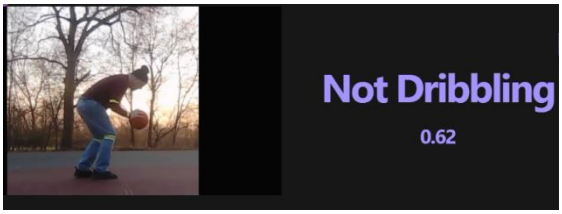
In our approach, we will apply Francois Chollet's universal workflow of machine learning design methods to our foul shot scenario. Using DNNs to create meaningful machine learning applications is an iterative approach, at first we must understand and formulate the problem our application is attempting to solve – we must know the domain. After resolving the use case, the next steps are to make decisions to find the correct data, extract the data, vectorize the data, and finally evaluate and train it to make predictions. A first pass of analyzing, evaluating and training also goes through initial evaluations and measurements of success. Before making deployment decisions, the need for changes or customizations to the machine-learning model may arise, making our design methods an iterative process of smaller experiments. Issues arising during these iterative experimental processes may deal with data scrubbing, or the business model needing a deeper dive, or baseline measurements needing adjustments. Design decisions regarding the use of pre-trained models, transfer learning, or further customizations also occur in an iterative approach (Chollet, Deep Learning with Python, 2018), (Cai et al, 2020), (Elgendy, 2020).



In trying to build an understanding of the use case in the first iteration, I noticed there is a certain repetitive approach to a basketball foul shot. (Appendix 5. **Initial Video Trials**). First the person steps up to the foul line, dribbles, sets, aims, shoots, and follows-through. As a designer in the Google ecosystem, there are some design playgrounds to use that attempt to capture these repetitive images. These design playgrounds are a first step to start testing different modeling ideas in a machine-learning environment. First we have the website TeachableMachine.withgoogle.com created by the Google Creative Lab in 2017 which "is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone." Next is ml5js.org, which uses Google's TensorFlow.js to allow machine learning in the browser with minimal dependencies. These two 'play spaces' are a development environment to see if a design idea is workable. The actual experiments and first steps to attempting to classify dribbling, aiming and shooting was difficult. The video model needed more training to recognize all three actions while the video was streaming. Further issues were predictions only dealing with classifications and not future forecasts, or time series analysis.

Teachable Machine uses transfer learning of Google's MobileNet architecture so you can classify your own images. The first iteration in design consists of taking a sample of about 100 images of dribbling, 100 images of setting, and 100 images of shooting, and then loading them into Teachable Machine. Teachable Machine trains the model, and then exports the weights and the model files, for deployment to our trial application. This allows the designer to see if the design concept has any chance to become a part of the overall project.



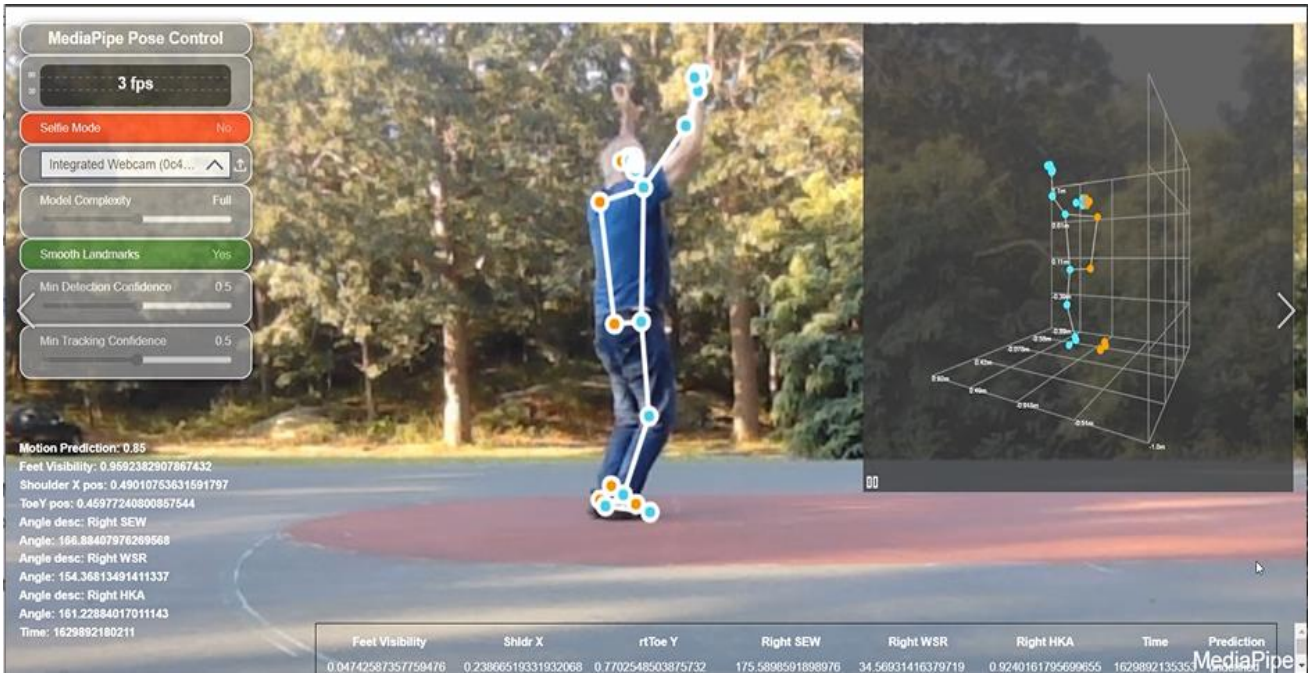


Examining the two images to the left, show the accuracy of our model is working, but needs more training and image data to increase the accuracy. Even if this first model does not yield the best model for the use case, some good lessons come from developing it. There is some excellent Python code found in the Keras community pages to scrape images frame by frame from an mp4 video (Keras, 2021). This code is in Appendix 1 under the title **Video Scraping Code**. Improving the accuracy of this model is possible by feeding the layered model more images of positive and negative results. Improving this iteration is not the direction to go. The focus of direction is to examine proper motion to send the basketball into the basket and to get feedback from our model to improve shooting accuracy. Our focus is on developing the motor skills in the shooting aspect of the mind-body experience. Once this is in our model, the addition of the action features will assure the person is following a repetitive process. Note, during the calibrations and scraping of images from an mp4 streaming video, changing images to greyscale saves memory and computational power. The code to assist in turning color to greyscale is in Appendix 1 under the Title **Greyscale**. The images above are part of a browser application, which is fully functional and important main portions of the code are in Appendix 1 under the Title **Foul Shot Action Classifier Code**. The focus is to develop the section of the app to demonstrate how the latest DNN models improve human motor skills with the latest in edge architecture.

From a purely semantic point of view, the first iteration attempt only classifies nouns and not verbs like shooting or aiming or setting. Looking semantically at the use case, the problem may sound something like what follows. The human steps-up to the foul line, dribbles until they feel comfortable, then coils in a set position to bend their knees, while simultaneously bending their shooting arm at an angle, to next push with their legs and rotate their upper arm at the correct speed and force to release the ball from their hand, at the optimal angle, to project the ball into the basketball hoop. Yes, this is an awkward sentence, but it shows, from a semantical point of view, that our solution to the problem for motion is full of verbs and not nouns – and it is also a very complicated human motion problem. This section points out the complexities to explore with the latest DNN models built by Google. Using Teachable Machine and similar design playgrounds start the iterative design and development process and show how JavaScript works with the browser and edge architecture to work with human motion. The next sections show the progressing iterative steps to customize and use well-developed machine learning models that are pre-trained and scalable to our examination of human motion with models developed with computer vision CNNs and the skeletal pose models. It is not only important to iterate through the model we may use, but a re-examination of the business use case and domain should be looked at again to assure a machine learning model will help.

2.2 The Foul Shot Mirror – Interface Design

The Foul Shot Training Mirror has a dual purpose. The first purpose allows an individual to extract data from the interface. The second purpose is to use the data extracted from the CNN to train a RNN persistent bit file to make predictions for the interface. The illustration below shows the completed Foul Shot Training Mirror interface. Section 3.1 describes the data flow from video streaming to output.



In the lower left is the output of motion predictions, key points, and key angles measured in real-time. The shooter sees this output on a screen projector in their line of sight to allow for adjustments to their motor skills. The predictions are of the shooter's most optimal shooting mechanics, which allows immediate feedback to adjust their body mechanics to become a more consistent free throw shooter. Note in the bottom right is the output of key points and key angles that are collected every 90 milliseconds. The data from this table is then used to train the persistent RNN model. The CNN model

outputs two colors, orange and aqua. The orange is the left side of the body and the aqua is the right side of the body.

Chapter III

Data Collection, Processing and Training

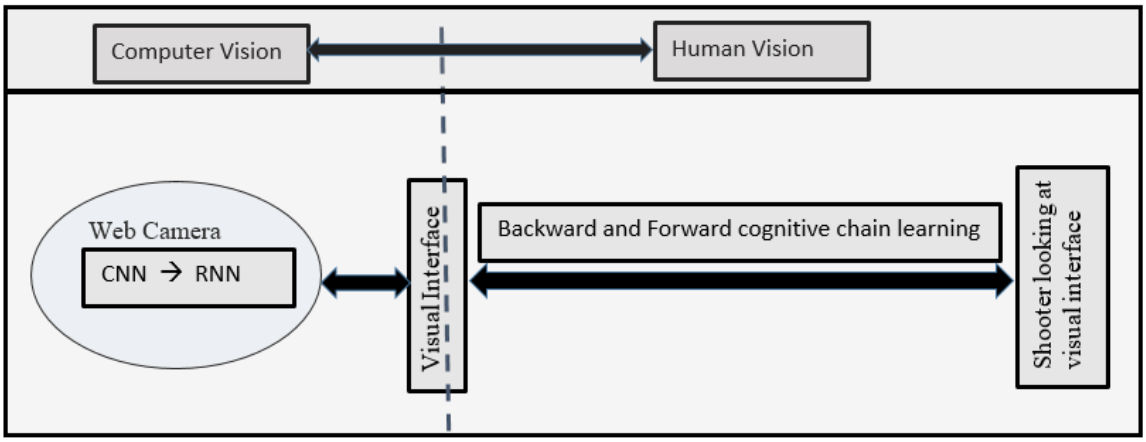
Section 3.1 discusses the streaming video data flow from the web camera into the CNN and RNN to the output of the visual interface for the person shooting the foul shot. Section 3.2 discusses the importance of becoming an expert in the field of study to understand the data under analysis. The RNN algorithms proposed need help understanding important pieces of the data. Understanding the data is essential to reduce the number of rules before the labeling and training process begins. As a deep learning engineer, reducing the hypothesis space is an essential task; otherwise, the RNN spends an unreasonable amount of time finding a solution.

3.1 Video Streaming Data

Video streaming data flows from the webcam into the web browser with JavaScript gluing all the pieces needed to create the visual interface. MediaPipe Pose is the next important step in the data flow process. The webcam pipes the data images into the CNN inside of MediaPipe's Pose model where 33 skeletal points are determined. Of the 33 points only a subset are needed to create the output for the Foul Shot Mirror. The subset is determined from the observations and experiments discussed in the following chapters and sections. The subset of data determines the right side of the body verses the left side. It also determines significant angles used by the shooter to project the ball into the basket. As outlined in the experiments section, data is collected to feed an RNN using TensorFlow.js. TensorFlow.js is used because we pipe in the predictor function to

read our saved model stored as a bit file. The bit file is created with Python Keras and converted to JavaScript with TensorFlow's model converter. Data is then output to the interface in the form of predictions, which are dribbling, pre-set, set, and a confidence level of success for the basketball foul shot.

The diagram below shows the instant feedback loop. The instant feedback loop enables the computer vision to interact with human vision, therefore increasing human cognitive awareness of fine motor skills.



The above interaction and feedback loop occurs within 3,000 milliseconds. The interactive sensory effect increases because of the tight feedback loop. The individualized RNN model to predict optimal motions steers the behavioral changes of motor skills through the cognitive influences of the mirror. As a result, the mirror-like interface creates a new way to study cognitive awareness of motor skills in the human body. An individual can work in private, on a team, or in a larger social network to enhance or study their motor skills, therefore enhancing and substituting the more traditional methods of learning fine motor skills.

From the experiments below, we determine the most significant key points from the skeletal model, and most significant angles for angular velocities to effect the success

of the foul shot. The purpose of this is to pick out the skeletal points that attribute to the optimal body mechanics of the foul shooter. Picking out the key points and understanding the data is important to lower the hypothesis space. Lowering the hypothesis space reduces the cost of learning the large number of combinations presented to the training of the RNN. By only presenting the RNN algorithm with solid data from our experiments and observations, we create a better predicting model by understanding the data.

3.2 Data for Motion Monitoring

After working with some pre-design models, it is time to take a harder look at our domain and the data to extract and resolve our machine learning motion detection dilemma. The first problem at hand is to design our application in such a way as to extract data from the shooter when they shoot a successful foul shot, secondly to provide feedback on how to correct body motions to improve motor skills and increase the percentage of a successful foul shot. The design must scale to monitor many varieties of individualized shooting mechanics, so careful design choices to extract features by using DNN models with the ability to normalize data between varying individuals is important. To understand our use case, our model will determine the important features from a webcam video analysis of a human shooting a foul shot. To get a better understanding of the domain, listen to the referenced you-tube video to understand how a professional basketball player approaches the foul line (NBA_TV, 2019). The reason why this is such a great video is because it represents a time before modern sports analytics started any discussions about DNNs working with image analysis. The

important points Red Auerbach makes in his interview with Rick Barry and Jamal Wilkes show that these proven basketball athletes have a sort of sixth sense. Our electronic personal basketball foul shot assistant will help the average person find this sixth sense, learn to fine tune their motor skills, optimize their motion, and become more successful at the foul shot.

In the video, Red asks Rick Barry, "Why do you bounce the ball three times, not two, not four?" Rick's response is 'repetition,' it's his habit. In other words, it becomes part of his subconscious. If you listen closely, Rick even says, "I think it is repetition, it's the same way time after time, so that it becomes something that just gets embedded into your mind." This is why I believe part of the goal for our research is to raise cognizant awareness and to find ways to increase an individual's awareness of their mind-body relationship. A person can increase their cognizant awareness with the aid of the current technologies in Deep Neural Networks. This human-machine symbiosis will help humans learn optimal motions by strengthening their mind-body connection. The professionals in the video also discuss the softer touch and the backspin to help the likelihood of the basketball going into the hoop, which is a helpful bit of information to learn about the domain. So which way should you shoot? Both Rick and Red believe that you should shoot the way you feel most comfortable and have the most confidence, and that is why this application will use each individual shooter's data output for training to personalize a shot and embed the repetitive steps into the user's repertoire. Next, Red compares the different style of shooting with Jamaal Wilkes. Red says, "If I ever had to teach his way, I'd die a horrible death." Red finishes off by rhetorically asking, "Is there a right way? I say no" Either way you shoot, it does not matter. "The answer is do what

is best for you. Do it the way you can make it. That's the name of the game. Relax, follow-through, but make it!" (NBA_TV, 2019)

For those of us who are not natural athletes, and our bodies do not naturally adapt to Red Auerbach's strategy of simply relaxing and following through, there are more formal studies of the mechanics of basketball (Marion Alexander et al, 2014). In addition to personalizing the foul shot assistant, we will take advantage of the formal mechanical studies to get a better understanding of the domain of the mechanics of shooting a foul shot to include these more formal interpretations within our model. Most studies indicate a repetitive repertoire is important, but the consistency of release angle, release timing, and minimal variability in body mechanics between shots attributed to the success of the foul shot. One cannot ignore the psychological impacts of stress during any human or competitive sporting event. In this experiment, the focus is on a practice shot in a non-game situation. There is room for future experimentation with DNNs for the effect of adding stressful situations to the human shooter, but we will not analyze it here. The benefit to setting a baseline will help to understand stress and anxiety on the motor skills of an individual in future studies (David J. Wright et al, 2018). In other studies, research is carried out to determine if it is possible that during the learning process in practicing a foul shot can increase the ability to acquire motor skills (F. MartijnVerhoeven et al, 2016), (Kearney et al, 2017). This is significant because if one can monitor and acquire motor skills to improve deficiencies, whether in sports, or for general health and happiness, it is a positive impact for the person and society. The significance of motion studies also influences applied sciences in the manufacture of compression garments

(Wong et al, 2020). Also, see Appendix 6 for **list of Common errors in Free Throw Shooting** (Marion Alexander et al, 2014).

Chapter IV

Skeletal Models and Experiments

Let us define our problem so far. We are given these fixed foul shot variables consisting of a basketball measuring nine inches in diameter, weighing twenty-two ounces, a basket measuring eighteen inches in diameter, a foul line fifteen feet away from a basket ten feet in the air(Lockard, 2021). Since these items represent our fixed variables, focusing the design of our machine learning algorithms is not here. The focus for the machine learning design is where the feature maps need creation, meaning the focus is on the human motion that sends the basketball over a fixed distance into the basketball hoop. This process emulates a standard physics problem of a catapult sending a projectile with a certain magnitude of velocity and angular velocity to reach a certain distance. If we were to set a mechanical catapult arm at the foul line, we could create an arm that would project a basketball at the proper angle and speed to reach a high degree of accuracy of scoring each time. This problem presents itself in the same way, but instead of a fixed mechanical arm, we have human perceptions, cognitive abilities, and adjustments to motor skills measured with The Foul Shot Mirror. In fact, traditional ways of a human learning a foul shot is by trial and error, by shooting a foul shot in a repetitive way, and changing their motor skills by using their intuition and moving their body to score a higher percentage of foul shots. The design of our machine learning will emulate the cerebral process of manipulating these body positions to optimize the motor skills by learning the features of a video stream of a person shooting a foul shot.

As discussed above, knowledge of the domain is important to pointing our machine learning algorithms at the correct features. As Redman points out in a recent Harvard Business Review article, practical experiences of the modeler with the data model are important for a successful implementation of your findings (Redman, 2018). By actually getting out and shooting foul shots, the modeler, while learning to advance certain motor skills, begins to formulate a machine-learning model. The experienced modeler knows the proper machine learning models to apply to the problem at hand, knows when to customize, and knows the systems necessary to bring the application to fruition. When the data modeler observes the learning process of how to throw the basketball into the basketball hoop from the foul line, they realize they are learning to adjust certain motor skills to change their human motions to affect the flight of the basketball. Human intuition applies these rules, using their mind to make their bodies enable the motions they believe will increase their chances to make the basketball fall through the hoop. In this machine learning design, we want our machine-learning model and algorithms to take the rules learned intuitively and help automate the learning process. Essentially, our intuition as to what makes a successful shot will result in labeling what makes a higher probability of a shot going into the basket, versus not getting to its destination. The ability to label our intuitive thoughts against the proper Deep Neural Network model makes the use of DNNs a natural progression for the symbiosis of human and machine integration.

Our model will take advantage of the idea of the catapult physics problem mentioned above and the skeletal DNNs used in Google's skeletal simulation models. The model to design will make a representation of the cerebral process

occurring within a basketball player routine while learning to shoot a foul shot. The model will use the DNN skeletal model to measure the angular velocities and positions of the foul shooter.

Set-up and aim

Shooting

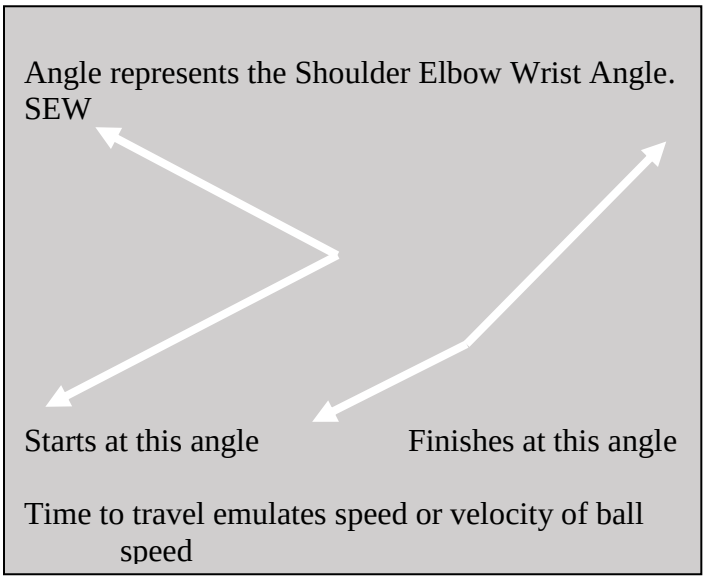
The two main actions of the shooter are to set-up and aim, then shooting by uncoiling the body to send the basketball on its way to the basket. There are three main angles to measure in our model. Each angle has a starting and ending point. Traditionally, we could measure the speed at which the angles change to determine linear and angular velocity to understand the optimal angles and time to change. However, the machine learning and digital video media with browser technology will measure this for us in near real-time. The output will provide feedback to the shooter to aid in their knowledge on how to alter their motor skills to increase the accuracy of their foul shot. The speed at which the angles change are the features our model will learn to determine optimal speed and angles to successfully shoot a foul shot.

4.1 DNN Skeletal Architectures

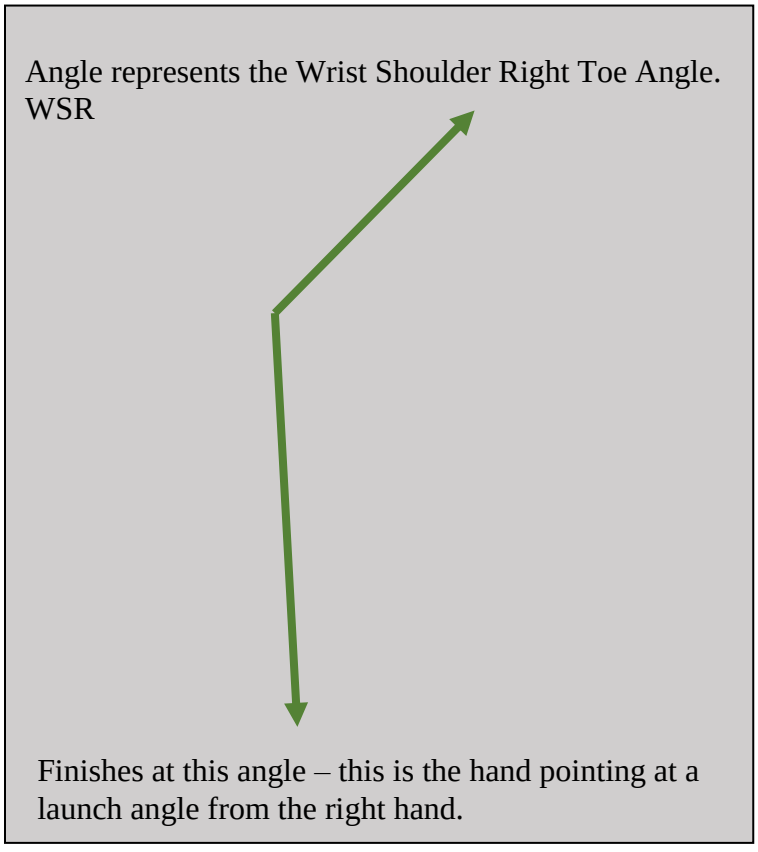
As you recall, the first iteration of our model focused on correct steps or actions leading up to and including the shooting of the basketball, but the next experiments will use pre-trained models. The pre-trained models are the skeletal models. This study uses the open source skeletal models from Google. The current models Google has are PoseNet, MoveNet, and MediaPipe Pose. The next iteration examines Google's PoseNet, and the important pieces of code are located in Appendix 1 under the Title **PoseNet**. This web application uses JavaScript, TensorFlow.js, and the PoseNet model from the pre-trained TensorFlow PoseNet models. The new customization is using code from a game demonstration that shows how to extract the key-point data from the skeletal PoseNet model(Rivera et al, 2020). The point here is that a machine-learning designer must be able to customize pre-trained models from Google's machine learning libraries to customize new ideas. The designer must be creative because every domain will not exactly fit into a pre-trained model, so as the iterative process plays out, the creative skills must take-over. Learning how to extract the proper key-point data is the biggest discovery in this iteration, this data extraction into the front-end index.html page allows us to follow the motions of the key-points in relation to time measured in milliseconds. The projection of the seventeen key-points to skeletal points in a streaming video is the key to measuring motion of the human body. Projecting the key-points onto the human body with deep neural network based image animation with JavaScript delivers near real-time results in the browser. The JavaScript code allows you to change the model and use the MobileNet or ResNet architecture, while also changing the output stride of the CNN architecture. Further, this model allows you to adjust the depth of the

layers in the transfer learning of the MobileNet CNN. Current practices suggest that a multiplier of 0.75 is a sufficient depth in identifying the key-points onto the human body to create an accurate pose. The results in this iteration were not impressive due to latency issues with the PoseNet skeletal model. This led to more research and to a finding of a better skeletal model, which has 33 points and a software release in December of 2020. The new skeletal model is Google's MediaPipe Pose and using it in the third iteration resolves the latency issue.

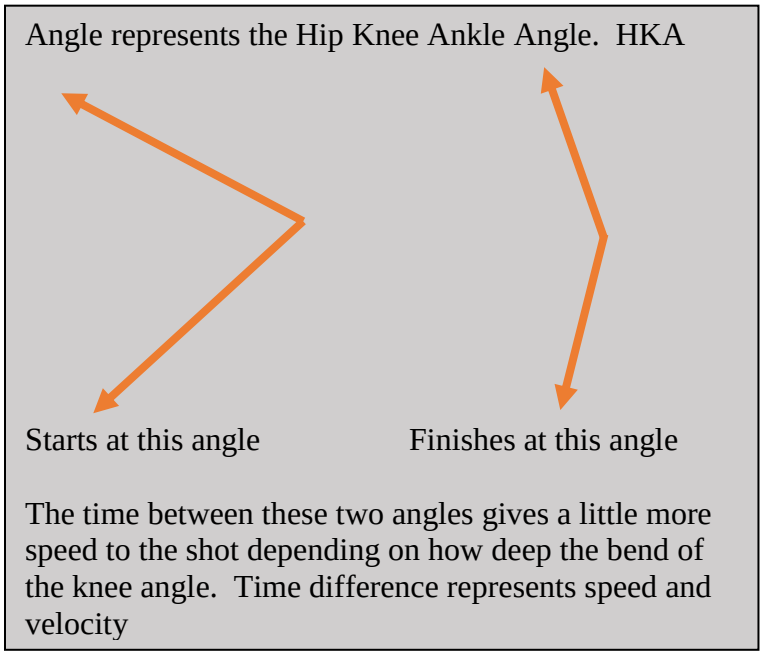
The third iteration brings Google's MediaPipe's Pose to the forefront. In review of the second iteration, we discovered the value of extracting the skeletal points from a live video stream. The problems with PoseNet and MoveNet is latency in drawing and extracting the key-points in a near real-time manner. Adding more customizations to the PoseNet model increases latency and hampered real-time progress. As noted above, the two main actions of the shooter are the action of 'set-up and aim' and then the act of 'shooting.' This breaks down to a physics type of problem, similar to measuring the forces and angles of shooting a projectile a certain distance.



These drawings to the left are depicting the three angles in the diagrams above - 'Set up and Aim' and 'Shooting.' The white angle above represents the Shoulder, Elbow and Wrist Angle in the first diagram. (SEW)



The next angle, depicted by the green arrow above, starts in the set position and its final position is illustrated to the left, it moves from start to finish over a certain period, therefore exerting a certain speed or velocity to the ball. Releasing the ball creates the shooting angle and projects the ball toward the basket. The green arrow above represents this with labels in the diagram as the Wrist, Shoulder, and Right Toe Angle. (WSR).



The angle in orange in the above diagram represents the Hip, Knee, Ankle angle. (HKA) The force projected by this angular velocity acts as extra force to project the ball to the correct distance. The deeper the bend and faster the push with the legs, then the more distance the ball will travel.

4.2 BlazePose architecture

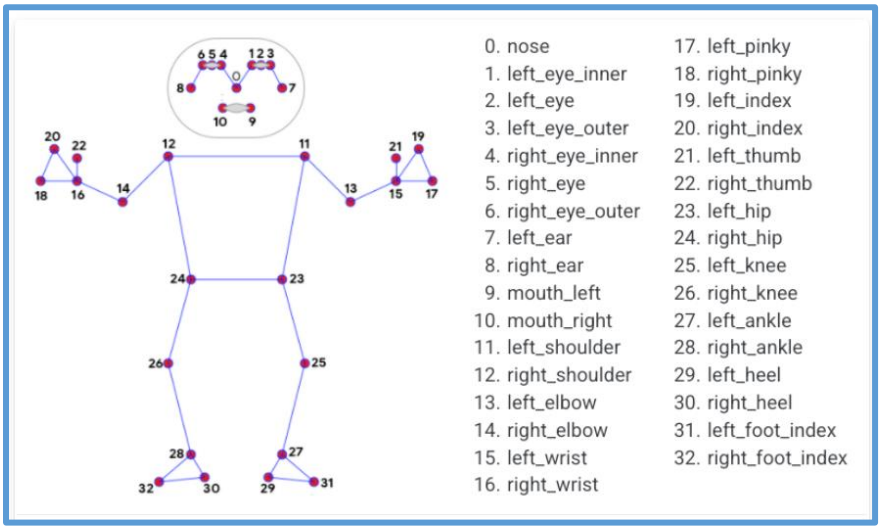
On Thursday, December 10, 2020, Google announced the release of its MediaPipe Pose Skeletal model with 33 pose estimation points for desktop, Android, and iOS (Ivan Grishchenko and Valentin Bazarevsky, 2020). The new MediaPipe BlazePose APIs are ready-to-use for the web in JavaScript. This architecture offers toe and hand Landmark points that will be useful in this basketball free throw analysis. The MediaPipe BlazePose technology offers near real-time machine learning in the browser. The architecture works best with few people, works in near real-time, uses mobile, is web-friendly and can work accurately at a distance of 19.50 feet. The unique architecture of the machine learning Landmark detection is that it uses a regression with heat-map supervision, deep integration with TensorFlow Lite on mobile/IoT for full HW acceleration (CPU, GPU, and Edge TPU) (Brunham, 2021).

See the BlazePose Skeletal Diagram below.

The actual points to draw the angles from the model map to the diagram below.

Angle abbreviation	Description	BlazePose Landmark and Description	BlazePose Landmark and Description	BlazePose Landmark and Description
SEW	Shoulder, Elbow, Wrist angle	12. Right Shoulder	14. Right Elbow	16. Right Wrist
WSR	Wrist, Shoulder, Foot angle	16. Right Wrist	12. Right Shoulder	32. Right foot index
HKA	Hip, Knee, Foot angle	24. Right Hip	26. Right Knee	32. Right foot index

4.2.1 BlazePose Skeletal Diagram



The Landmark Diagram is sometimes referred to as the skeletal model. This model has 33 pose Landmark points Pose - MediaPipe (google.github.io)

4.3 Extracting Angular Velocity from MediaPipe Pose

In Google's MediaPipe site, there is a Web Demo link of BlazePose JavaScript code in CodePen (CodePen-Google, 2021). The Web Demo code is JavaScript, and after examining the code and API of the BlazePose model, customizations to the code are

necessary. See Appendix 1 under the Title **BlazePose customization**. To properly extract the data and measure the angles, adding a formula to determine an angle and a time output in milliseconds is necessary to measure angular velocity. To determine an angle, we only need three points, and there is sample code on MediaPipe' s site, for proper customization to the code in the JavaScript file shown in Appendix 1 under the Title **Angular Calculation**. The final interface is illustrated below.

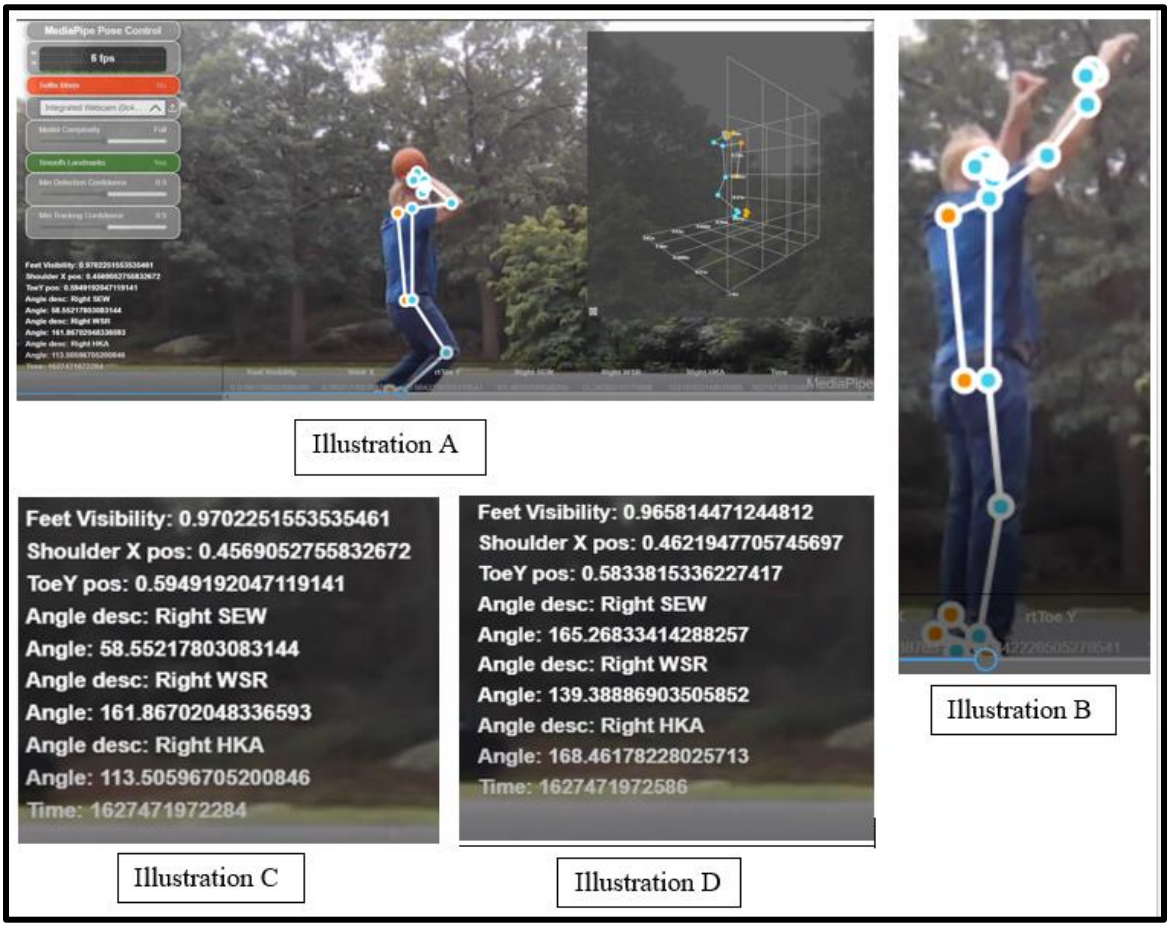
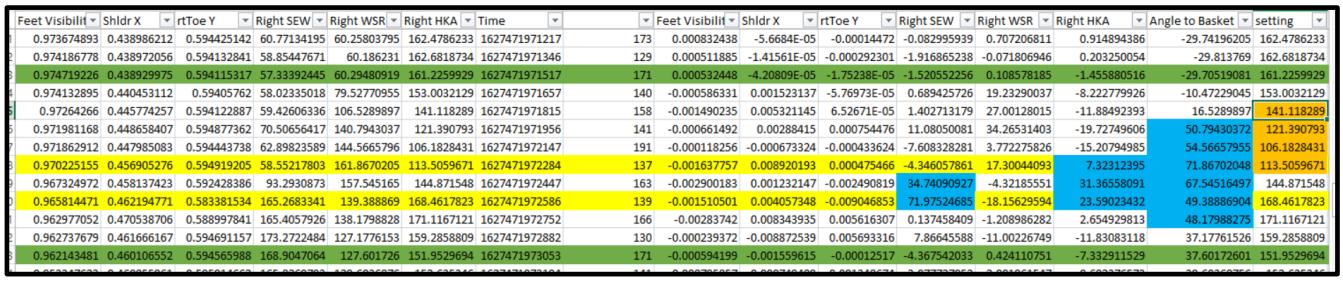


Illustration A is the data extraction web page. The data table in the lower left of illustration A is in Illustration C, and represents the angles in the pose diagram in illustration A at the time 1627471972284 milliseconds. Illustration B is the pose position at the time 1627471972586 milliseconds and the output is in illustration D. The data table for each millisecond representing one shot is below in the Data Output Diagram.

4.3.1 Data Output Diagram



Feet Visibili	Shldr X	rtToe Y	Right SEW	Right WSR	Right HKA	Time		Feet Visibili	Shldr X	rtToe Y	Right SEW	Right WSR	Right HKA	Angle to Basket	setting
0.973674893	0.438986212	0.594425142	60.77134195	60.25803795	162.4786233	1627471971217	173	0.000832438	-5.6684E-05	-0.00014472	-0.082995939	0.707206811	0.914894386	-29.74196205	162.4786233
0.974186778	0.438972056	0.594132841	58.85447671	60.186231	162.6818734	1627471971346	129	0.000511885	-1.41561E-05	-0.000292301	-1.916865238	-0.071806946	0.203250054	-29.813769	162.6818734
0.974719226	0.438929975	0.594115317	57.33392445	60.29480919	161.2259929	1627471971517	171	0.000532448	-4.20809E-05	-1.75238E-05	-1.520552256	0.108578185	-1.455880516	-29.70519081	161.2259929
0.974132895	0.440453112	0.59405762	58.02335018	79.52770955	153.0032129	1627471971657	140	-0.000586331	0.001523137	-5.76973E-05	0.689425726	19.23290037	-8.222779926	-10.47229045	153.0032129
0.97264266	0.445774257	0.594122887	59.42606336	106.5289897	141.118289	1627471971815	158	-0.001490235	0.005321145	6.52671E-05	1.402713179	27.00128015	-11.88492393	16.5289897	141.118289
0.971981168	0.448658407	0.594877362	70.50656417	140.7943037	121.390793	1627471971956	141	-0.000661492	0.00288415	0.000754476	11.08050081	34.26531403	-19.72749606	50.79430372	121.390793
0.971862912	0.447985083	0.594443738	62.89823589	144.5665796	106.1828431	1627471972147	191	-0.000118256	-0.000673324	-0.000433624	-7.608328281	3.772275826	-15.20794985	54.56657955	106.1828431
0.970225155	0.456905276	0.594919205	58.55217803	161.8670205	113.5059671	1627471972284	137	-0.001637757	0.008920193	0.000475466	-4.346057861	17.30044093	7.32312395	71.86702048	113.5059671
0.967324972	0.458137423	0.592428386	93.2930873	157.545165	144.871548	1627471972447	163	-0.002900183	0.001232147	-0.002490819	34.74090927	-4.32185551	31.36558091	67.54516497	144.871548
0.965814471	0.462194771	0.583381534	165.2683341	139.388869	168.4617823	1627471972586	139	-0.001510501	0.004057348	-0.009046853	71.97524685	-18.15629594	23.59023432	49.38886904	168.4617823
0.962977052	0.470538706	0.588997841	165.4057926	138.1798828	171.1167121	1627471972752	166	-0.00283742	0.008343935	0.005616307	0.137458409	-1.208986282	2.654929813	48.17988275	171.1167121
0.962737679	0.461666167	0.594691157	173.2722484	127.1776153	159.2858809	1627471972882	130	-0.000239372	-0.008872539	0.005693316	7.86645588	-11.00226749	-11.83083118	37.17761526	159.2858809
0.962143481	0.460106552	0.594565988	168.9047064	127.601726	151.9529694	1627471973053	171	-0.000594199	-0.001559615	-0.00012517	-4.367542033	0.424110751	-7.332911529	37.60172601	151.9529694

The purpose of the above iteration is simply to verify if the output data is useful to build and train our DNN model. Notice the two yellow highlighted lines above represent the two pose positions. The accuracy is extraordinary; see that our model extracts three data points in 439 milliseconds. The orange highlighted column entitled setting is the Right HKA angle, which stays at about a 160-degree angle until the legs bend to decrease the angle from 160 to 113 degrees. This detailed examination of features that the human motion creates represents the mind-body connection to optimize the projection of the ball into the basketball hoop. The near real-time feedback loop allows the human to adjust the positioning of their body to 'help human beings acquire, represent, and integrate knowledge' (Campero et al, 2018). The important item to note in the column titled Angle to Basket and highlighted in blue, is the average angle to the basket for the angles in blue highlight is fifty-seven degrees. A cross reference to 'Mechanics of the Basketball Free Throw' shows a release angle is optimal at a high point and measures about fifty-five degrees (Marion Alexander et al, 2014). The fascinating point our mirror-like personal basketball assistant shows, is the feedback is immediate, allowing a backward or forward chaining learning process to occur through machine learning technology. The result of this shot was a success. These calculations are great, but we want the machine to

determine all of these feature map relationships for the predictor variables. So first, let us follow Chollet's model and create a commonsense baseline to see if our machine model will be better than our commonsense classical programming approach.

Chapter V

Baseline Models

The actual data in Data Output of section 4.3.1 above, is only the first seven columns, the additional columns are transformations to test our model assumptions to show the application is extracting useful data. The data, from a commonsense viewpoint is useful, and the programming in a manual fashion will give us a baseline to test our machine-learning model that we will create from this output. This commonsense approach outlined in 'the universal workflow of machine learning' is an identifiable way to measure the success of the trained model from our output (Cai et al, 2020). Our goal is to develop a model to beat our commonsense baseline, and to see if the machine-learning model is more accurate. This commonsense baseline programming approach is a more traditional non-machine learning approach, using data extracted from a pre-trained model. The concept behind the machine learning algorithm approach for this domain is the algorithm picks up dribbling when the angle illustrated in the diagram below, marked in green drops down to less than a twenty-five degree angle. Unlike the classification model developed with the Teachable Machine design tools, the skeletal model is universal to different size skeletal frames, changes in scenery, and changes in people of different shapes and sizes. In the skeletal model, each person will have these same 33 data points, and the model will not need to train itself to ignore irrelevant items. This shows the importance of a domain expert's knowledge to dig into the data and extract items that are helpful to the machine learning training set. The machine learning algorithms will take many more parameters while analyzing many relationships. It is the

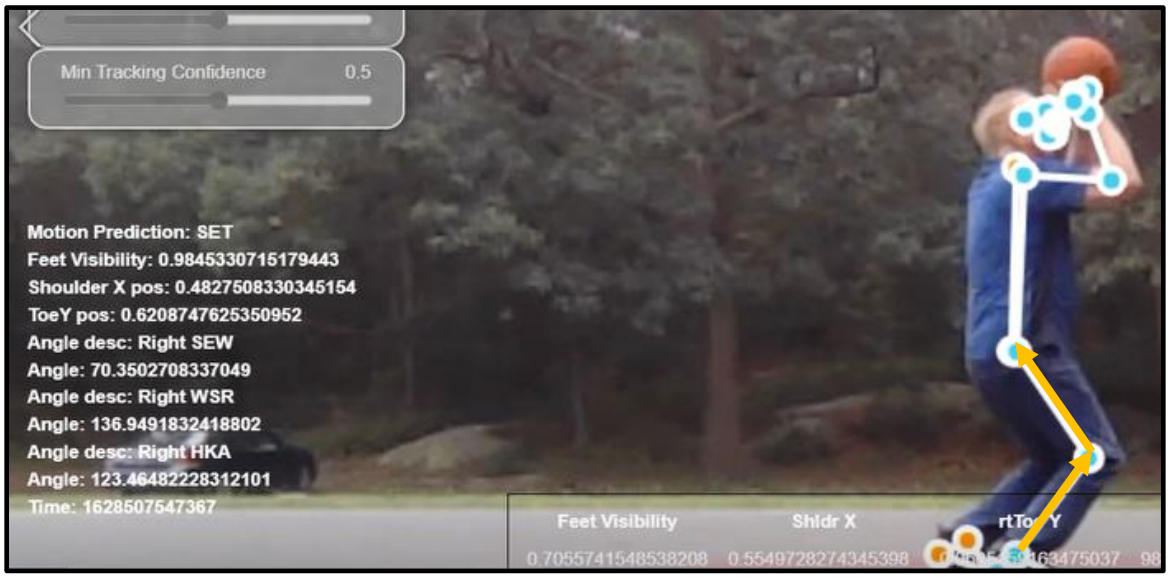
designer's responsibility, in a supervised learning scenario, to give the model relevant scrubbed data.

5.1 Three Key Positions

See how, in these three key positions, the three angles; green, orange and red change relative to time. Their change in position over time is a measure of angular velocity. The assumption is, if the angles and time feed into a machine-learning model, the training of the model will yield the parameters necessary to predict the necessary motions to shoot a basket at a 90% success rate.

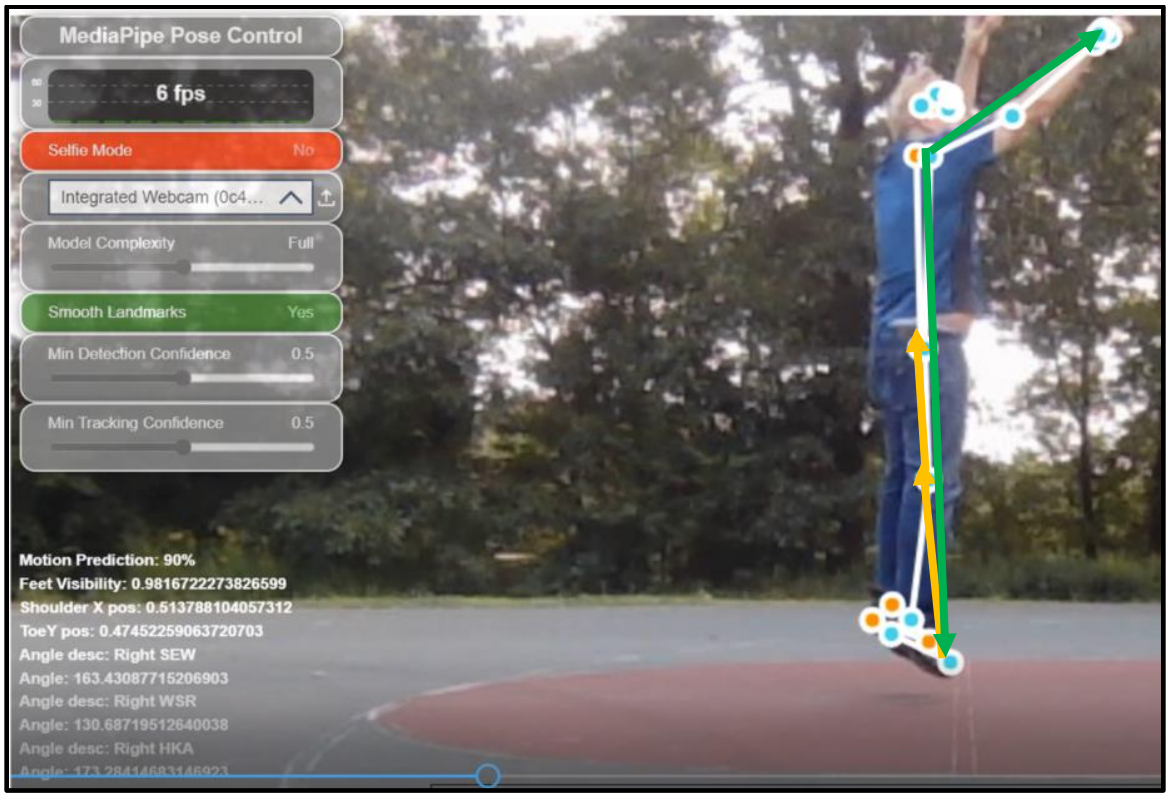
5.2 Green Angle

The green angle (WSR) between wrist, shoulder and right toe is less than 25.5 degrees so the Motion Prediction yields ‘Dribbling.’



5.3 Orange Angle

The orange angle (HKA) is less than 143 degrees indicates a Motion Prediction of SET.



5.4 Green and Orange Angle changing over time

The green angle (WSR) is greater than 130 degrees, and the orange angle (HKA) is greater than 160 degrees - indicating a 90 % chance of scoring.

Next, let us apply the RNN machine learning algorithms to the data output from the above descriptions of the Pose skeletal model. We will compare the machine-learning model to the commonsense baseline approach.

Chapter VI
CNN and RNN Models

What we have so far is a model pushing out data from a pre-trained Pose model. The pre-trained Pose model is the creation of Google Research, developed over the span of several years, delivering the data, low latency and privacy we need. Our challenge is to customize this well engineered model to avoid having to retrain our own skeletal model because it will be very expensive in time and resources. The commonsense baseline, in the above chapter, creates prediction outputs using classical style programming to transform the output data. Classical programming means the programmer enters the control statements for the conditions of the angles to output the predictions. See Appendix 1 under **Classical Style Programming** to see the control statements delivering the output of the motion prediction variable in iteration two above. It is impossible to use classical programming to customize all the possible features represented from the output of MediaPipe's Pose skeletal model. In addition, it is not feasible to reprogram the application for each new basketball shooter's style. What is feasible is for the machine to take the machine-generated data and create a machine-learning model. The model it creates is a representation of the angular velocity combinations to yield a prediction of the basketball going into the basket. We also want it to predict when the person is 'dribbling', in the 'preset' position, and the 'set' position.

As discussed above, we know that a modern machine-learning model will use many thousands of parameters, neurons, and features to solve our prediction problem (Elgendy, 2020). The question to answer now is which machine learning model will best describe our data output. Since our angular velocity representation relies on time series

and needs a memory state of the previous shooting position, we choose a deep learning model from the Recurrent Neural Network family of models. There are several Recurrent Neural Network models to choose from, namely RNN, LSTM, and GRU (Chollet, Deep Learning with Python, 2018) (Cai et al, 2020). Data collection is very important, and good data is the difference between a good and bad model, otherwise model performance is low and the iterative process of development begins again (Appendix 2 **Application and Data**).

Using Python and Keras (Keras, 2021) to build our persisting models we use a python converter function to convert the h5 format into a model in json format. Since our customization is using python, a conversion to a JavaScript json file format is necessary to use in our MediaPipe Foul Shot Mirror application (TensorFlow, 2021) (API_TensorFlow, 2021). Note we are using a TensorFlow script tag to pull the TensorFlow.js API into our model. Now our model is using MediaPipe APIs and TensorFlow.js APIs, where MediaPipe is controlling the BlazePose Model and TensorFlow.js is controlling the RNN model predictions with data output from the BlazePose skeletal model. See Appendix 1 under **TensorFlow Model Predictor** to see where prediction generates from the weights and model topology of the json file.

Chapter VII

Results and Discussion

7.1 Choosing a Model

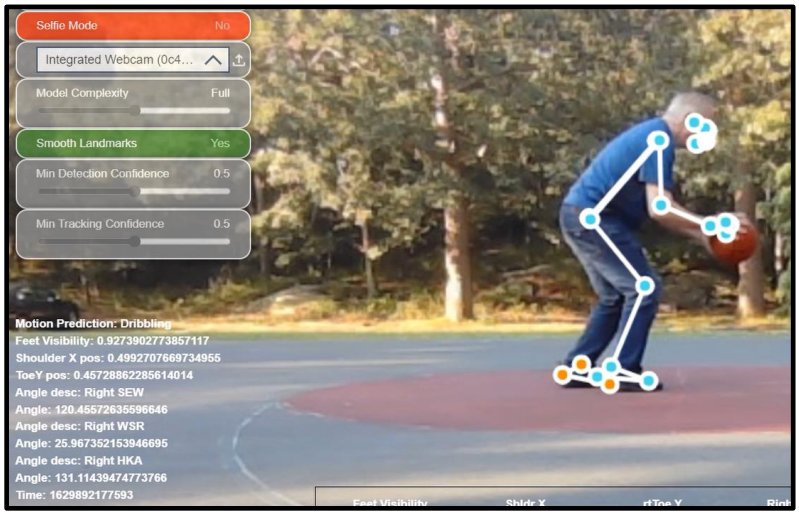
Taking the data from the skeletal model and labeling it to train with an RNN model is part of the process of hyper-parameter optimization. "Unfortunately there is currently no definitive algorithm that can determine the best hyper parameters given a dataset and the machine-learning task involved." (Cai et al, 2020) We look at three deep learning models to find the Gated Recurrent Unit (GRU) with Dense Layers performing better than the Long Short Term Memory (LSTM) and the Recurrent Neural Network with conv1d (Chollet, GitHub Repository, 2018), (Cai et al, 2020), (Geron, 2019). See Appendix 7 **Summary of Model Layers**. There is a plethora of research to compare different models with different data sets, and there are a few interesting studies to build our intuition on the subject. One research study examines GRU vs LSTM to discover that GRU is just as powerful as LSTM (Chung J. G. et al, 2014). Another study shows a GRU performs better than an LSTM under certain types of content. The study goes on to suggest that GRU outperforms LSTM when the sample size is small (Gruber et al, 2020). Intuitively, a basketball foul shot sequence pattern is not a long sequence; the significant parts of the foul shot occur in the sequence {Dribbling, Pre-Set, Set, Confidence} and usually occurs in less than 3,000 milliseconds. Therefore, this short sequence along with a smaller dataset may contribute to the better results of the Gated Recurrent Unit over the other models. The main goal of the Foul Shot Training Mirror is

to show RNNs not only apply to predicting the next word in a sentence, but also predict the next result in a series of motions.

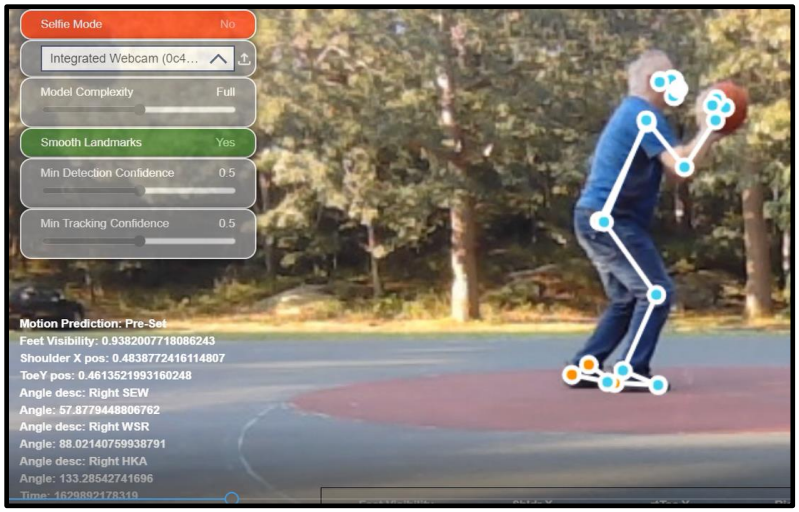
Therefore, the GRU model converts to persist in json format and is now the predictor for the Foul Shot Training Mirror application and testing begins at the basketball court. The set-up for the webcam is on a chair about eighteen inches off the ground, fourteen feet away and perpendicular to the foul line. The webcam must have the entire body in its view; have the selfie mode on, and pointing toward the shooting side of the person shooting the foul shot (Appendix 8 – **Set-Up Instructions**). The example shown is designed for a right-handed shooter. In all, the substitution of the 'classical programming' control statements with the RNN deep learning model works better than the commonsense baseline from iteration two above. The illustrations below represent a sequence lasting 2,618 milliseconds where the GRU successfully identified dribbling, pre-set, set, and gave the motion sequence an 85% chance of making a basket. The basketball shot did go into the basket to score the foul shot. Showing the GRU is able to predict the next result in a series of motions. The GRU model does perform better than the commonsense baseline proving that our deep learning model is the direction to go and is the cost effective alternative. Knowing it is cost effective, we can expand our experiments with more data trials and more people in future iterations. This will significantly increase our dataset and make our model even more accurate.

A personal observation after analyzing my results from the Foul Shot Training Mirror is a cognitive awareness of my body mechanics. It showed me, in a symbiotic way, to change my focus and to bend my knees a little deeper, push hard with my legs

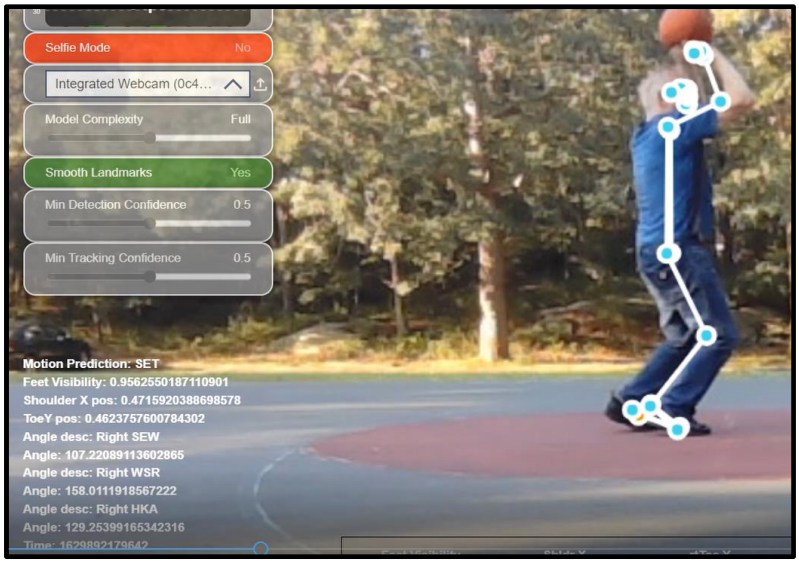
and point my hand to a higher position pointing at the top of the backboard, then the ball would go in and the confidence score rises. This cognitive action is the result of watching the Foul Shot Mirror and then changing my fine motor skills to affect my body positions. The GRU is performing better than the baseline, because when running after the ball, or dribbling away from the foul line the Motion Prediction remains 'Empty'. The commonsense baseline method will return predictions regardless of where I am standing, and return predictions if pretending to shoot the basketball. The deep learning model is not only successfully predicting the sequence, but we also were able to add the 'Preset' motion as part of the predictions. The fact this deep learning model works, shows the flexibility for new individual shooting styles to train their own personal shooting data. Below are screenshots of live streaming webcam video while the Foul Shot Training Mirror is running.



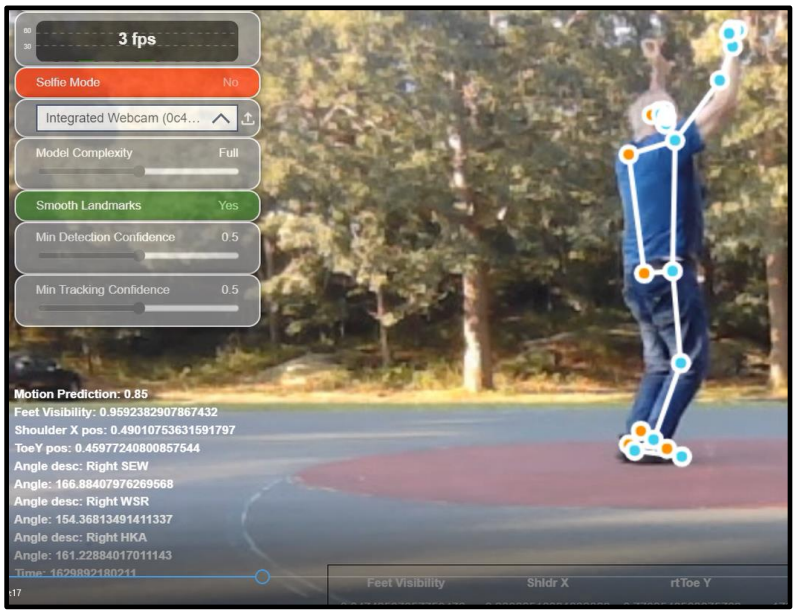
Dribbling at 1629892177593 milliseconds



Pre-Set at 1629892178319 milliseconds



Set at 1629892179642 milliseconds



Predicting 85% at 1629892180211 milliseconds

7.1 Usability

It is easy to see basketball gymnasiums may make this part of their equipment because the cost of this technology is low and creates an excellent learning and social atmosphere to learn physical skills. In the past only professional teams or elite teams had exclusive use of this type of training and technology. Tools like the Foul Shot Training Mirror developed with deep neural networks create a significant impact to allow better quality tools to a more diverse group of individuals. The diversity will affect a coach and player relationship, where the coach gives instructions after reviewing the images of the training mirror and then the individual can go work on their own and protect the privacy of results. By bringing anyone together to learn and analyze each other's technique of shooting helps set the direction to continue the research of Human Computer Interaction into the next generation. The ability of the tool to work privately and with low latency makes it a socially significant tool of the future. These new types of interactions create even more data and now even more ways to analyze, predict and innovate human motions.

7.2 Scaling to Other Basketball Shots

Now that the groundwork and baseline are set for analyzing the individual foul shot, the next iterative step is to examine the three point shot. The same type of experiments and iterative approach will work with many other shooting situations. Adding another geospatial variable as another predictor to determine where a person is on the court is necessary, but as proven, the deep learning algorithms will output the proper prediction model. Expanding the iterative process to other shots, and

shots with defenders, perhaps adding more predictors to identify a defender is also a natural next step. Intuitively, we know if a person is defending the shooter, then it will lead to a lower success rate. Again, the same iterative process works, but creativity must abound while designing the extra complexities. Now with more data, other tools will come to fruition, for instance, with the different types of shots in our model, the training mirror becomes an excellent scouting tool to recognize shooters with slightly less than perfect shooting styles. A top athlete with slightly less than perfect technique, while not standing out with high scoring percentages, will be the diamond in the rough, and easily recognizable for the Foul Shot Training Mirror and the scout trained to use it. A trained scout can use this tool to recognize a slight change in technique will raise the quality of the player. Using the tool to maintain performance is also a great benefit, because when the athlete begins to reduce the accuracy of their shot, they can review and compare past data to adjust their motor skills back to a prior state.

Chapter VIII

Conclusion

"… *Imagine sixty years from today, personal robot assistants small enough and trained to guide proper motions of our bodies to perform amazing feats, and aid in motion maintenance as we age or become injured.*"

(Zielinski, 2021)

One of the aims of this research is to examine the best deep neural network models to make predictions about human motion and to teach fine motor skills to improve a basketball foul shot. The approach taken is hands-on, where we discovered the proper motions to become a better and more consistent foul shooter. By experimenting on the basketball court with a web camera, we discovered the proper motions essential to completing a successful shot. The discovery yielded a sequence of steps to include, dribbling, pre-set, set, and shooting. Next, we discovered the best models to use by applying a machine learning workflow to build the working prototype that we call The Foul Shot Training Mirror. The iterative approach led to the discovery of piping both MediaPipe's Pose and TensorFlow.js through the web camera. MediaPipe's Pose deep learning based image animation allows the extraction of key body points for three major body angles to measure while shooting the basketball. We discovered the Pose model accurately pulls data from the web camera to measure the time series sequence within 3,000 milliseconds. The Pose model did this accurately while at a distance of 18 feet from the shooter. Next, training the time series data with three RNNs led to the discovery

that the Gated Recurrent Unit (GRU) is the best predictor of the shooting sequence. In all, MediaPipe Pose with TensorFlow.js piped through the web camera and using a Gated Recurrent Unit (GRU) deep learning model to make predictions with the output from the skeletal model is the best deep neural network set-up to examine human motion and to teach fine motor skills. In other words, we discovered the best set up is the MediaPipe Pose CNN based deep learning model, which feeds skeletal simulation data into the TensorFlow.js predictor model, which then reads the GRU deep neural network model to make motion predictions. Further, this prototype dissolves the roadblocks of latency and privacy of individual data issues by making predictions and inferences entirely with local processing in the client browser. The discovery of these deep learning models enhancing motion detection, simulation, and cognitive awareness in a private environment all contribute to the advancements in the study of human computer interaction (HCI).

Applying this quality of mirror-like feedback applies to many modalities that require the refinement of fine motor skills. The mirror-like feedback also allows for immediate forward and backward chain learning, which is an important style of learning to improve fine motor skills. This prototype can scale with many other applications. For instance, applications to guide a professional or amateur athlete to learn a new skill, a physically impaired auto accident victim to recover from their injuries, an amputee learning to use a new prosthetic, an elderly patient to maintain proper walking motions and avoid an unnecessary fall, and even lifetime monitoring of proper body mechanics as humans age. The Foul Shot Training Mirror delivers real time information making a person more cognizant, both somatically and at a subconscious proprioceptive level.

Appendix 1.

Code Highlights

Video Scraping Code

```
import os
import cv2
saveTo = 'C:\\Users\\edzie\\Pictures\\Frames3'
path = 'C:\\Users\\edzie\\Pictures\\Source'
#def create_Frames_FromVideos():
for video in os.listdir(path):
   #print (video)
     # try:
      # Opens the Video file
   videoPath = os.path.join(saveTo,video)
   print(videoPath)
   cap= cv2.VideoCapture(videoPath)
   i=0
   while(cap.isOpened()):
      ret, frame = cap.read()
      if ret == False:
         break
      #savePath = os.path.join(save_to, 'frame'+str(i)+'.jpg'
      print(savePath)
      cv2.imwrite(os.path.join( 'C:/Users/edzie/Pictures/Frames2',
'frame'+str(i)+'.jpg'),frame)
      i+=1
    #   except Exception as e:
    #       pass
#create_Frames_FromVideos()
```

To add a row, hover your mouse to the left of one of the gray lines between or after rows; to add a column, hover your mouse above one of the gray lines between or

Greyscale

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import cv2
%cd C:\\Users\\edzie\\
%ls
```

```
image_color = mpimg.imread(‘fileName.jpg')
plt.imshow(image_color)
image_color.shape
image_gray = cv2.cvtColor(image_color, cv2.COLOR_BGR2GRAY)
plt.imshow(image_gray, cmap = 'gray')
image_gray.shape
```

Foul Shot Action Classifier Code

```
//First iteration ml5 showing DNN model use to make predictions
const startModelClassification = () => {
   console.log('ml5 version:', ml5.version);
   // Initialize the variables required
   const classifier = ml5.imageClassifier('./src/model_7/model.json',
modelLoaded);
   const img = document.getElementById('frameContainer');
//C:\Users\edzie\VideoScraping\src\model_DribNotDrib_6_2_2021
   // When the model is loaded
   function modelLoaded() {
      console.log('MobileNet has been loaded!');
      setInterval(() => {
         let dataurl = videoCanvas.toDataURL();
         img.setAttribute('src', dataurl);
         // Make a prediction with a selected image
         classifier.classify(img, (err, results) => {
            if (err) {
               console.error(err);
            } else {
               let best_result = results[0];
               console.log(best_result)
               document.querySelector('#label').innerHTML = best_result.label;
               document.querySelector('#confidence').innerHTML =
best_result.confidence.toFixed(2);
            }
         });
      }, 200)
   }
}

const startModelClassification = () => {
   // ml5
   console.log('ml5 version:', ml5.version);
   // Initialize the variables required
   const classifier = ml5.imageClassifier('./src/model_7/model.json',
modelLoaded);
```

```
            const img = document.getElementById('frameContainer');
          //C:\Users\edzie\VideoScraping\src\model_DribNotDrib_6_2_2021
            // When the model is loaded
            function modelLoaded() {
               console.log('MobileNet has been loaded!');
               setInterval(() => {
                  let dataurl = videoCanvas.toDataURL();
                  img.setAttribute('src', dataurl);
                  // Make a prediction with a selected image
                  classifier.classify(img, (err, results) => {
                     if (err) {
                        console.error(err);
                     } else {
                        let best_result = results[0];
                        console.log(best_result)
                        document.querySelector('#label').innerHTML = best_result.label;
                        document.querySelector('#confidence').innerHTML =
best_result.confidence.toFixed(2);
                     }
                  });
               }, 200)
            }
          }
```

PoseNet

```
          //Module index.js
            async function getPose() {
              const pose = await model.estimateSinglePose(video, {
              flipHorizontal: true,
             });

            drawKeypoints(pose.keypoints, MIN_CONFIDENCE, ctx);
            drawSkeleton(pose.keypoints, MIN_CONFIDENCE, ctx);

          //Module draw.js
                  inputs.push({partNumber: i, score: keypoint.score, x: x, y: y, part:
keypoints[i].part, time: time});
                  console.log('partNumber is ' + i + 'keypoints object is ' + keypoints[i].part +
'\n'
                  + 'score is ' + keypoint.score)

           model = await posenet.load({
            //model = await movenet.load({
            architecture: 'MobileNetV1',
```

```
            outputStride: 16,
            inputResolution: { width: SIZE, height: SIZE },
            multiplier: 0.75,

          });
```

BlazePose customization

```
   /* RIGHT side three points index */
   const shoulderIdx = 12;
   const elbowIdx = 14;
   const wristIdx = 16;
   const rfFootIdx = 32;
   const rHipIdx = 24;
   const rKneeIdx = 26
   const rAnkleIdx = 28
   /* RIGHT side three points index */

function getAngle(firstPoint, midPoint, lastPoint) {
   console.log('shoulder x is ' + firstPoint.x + 'and y is ' + firstPoint.y)
   let result = radians_to_degrees(Math.atan2(lastPoint.y - midPoint.y, lastPoint.x -
midPoint.x) -
      Math.atan2(firstPoint.y - midPoint.y, firstPoint.x - midPoint.x));
   result = Math.abs(result);
   return result > 180 ? (360 - result) : result;
}


/* calculate angle */
   const shoulder = { x: results.poseLandmarks[shoulderIdx]['x'], y:
results.poseLandmarks[shoulderIdx]['y'] };
   console.log('shoulder x is ' + shoulder.x + 'and y is ' + shoulder.y)
   // document.getElementById('shoulderx').innerHTML = `${shoulder.x}`;
   const elbow = { x: results.poseLandmarks[elbowIdx]['x'], y:
results.poseLandmarks[elbowIdx]['y'] };
   console.log('elbow x is ' + elbow.x + 'and y is ' + elbow.y)
   const wrist = { x: results.poseLandmarks[wristIdx]['x'], y:
results.poseLandmarks[wristIdx]['y'] };
   console.log('wrist x is ' + wrist.x + 'and y is ' + wrist.y)
      /*Begin calculate shooting angle  */
   console.log('right foot is ' + rfFootIdx)
   const rfFoot = { x: results.poseLandmarks[rfFootIdx]['x'], y:
results.poseLandmarks[rfFootIdx]['y'] };
   console.log('Right foot x is ' + rfFoot.x + 'and y is ' + rfFoot.y)
```

```
   /*End calculate shooting angle  */
    /*Begin calculate catapult angle  */
   console.log('right hip is ' + rHipIdx)
   const rHip = { x: results.poseLandmarks[rHipIdx]['x'], y:
results.poseLandmarks[rHipIdx]['y'] };
   console.log('Right foot x is ' + rHip.x + 'and y is ' + rHip.y)

   console.log('right knee is ' + rKneeIdx)
   const rKnee = { x: results.poseLandmarks[rKneeIdx]['x'], y:
results.poseLandmarks[rKneeIdx]['y'] };
   console.log('Right foot x is ' + rKnee.x + 'and y is ' + rKnee.y)

   console.log('right ankle is ' + rAnkleIdx)
   const rAnkle = { x: results.poseLandmarks[rAnkleIdx]['x'], y:
results.poseLandmarks[rAnkleIdx]['y'] };
   console.log('Right foot x is ' + rAnkle.x + 'and y is ' + rAnkle.y)
   /*End calculate catapult angle  */
   const angle = getAngle(shoulder, elbow, wrist);
   const shangle = getAngle(wrist, shoulder, rfFoot)
   const catangle = getAngle(rHip, rKnee, rAnkle)
```

Angular Calculation

```
        function radians_to_degrees(radians) {
           const pi = Math.PI;
           return radians * (180 / pi);
        }

        function getAngle(firstPoint, midPoint, lastPoint) {
           console.log('shoulder x is ' + firstPoint.x + 'and y is ' + firstPoint.y)
           let result = radians_to_degrees(Math.atan2(lastPoint.y - midPoint.y, lastPoint.x
- midPoint.x) -
              Math.atan2(firstPoint.y - midPoint.y, firstPoint.x - midPoint.x));
           result = Math.abs(result);
           return result > 180 ? (360 - result) : result;
}
```

Classical Style Programming

```
           if (visibility >  0)
           {
        // shangle = getAngle(wrist, shoulder, rfFoot)
```

```
        if (shangle < 25.5)
        {
           console.log('Dribbling')
            msgAction = 'Dribbling'
        }
    // catangle = getAngle(rHip, rKnee, rAnkle)
        if (catangle < 143)
        {
           console.log('SET')
           msgAction = 'SET'
        }
    // shangle = getAngle(wrist, shoulder, rfFoot)
    // catangle = getAngle(rHip, rKnee, rAnkle)
           if ((shangle >130) && (catangle > 160) )
           {
              console.log('90%')
              msgAction = '90%'
           }

     }
```

TensorFlow Model Predictor

```
    // load the model
    async function loadModel(path){
       console.log("Model loading in progress from ".concat(path));
       const model =  await tf.loadLayersModel( path);
       console.log("Model Loaded Successfully");
       return model;
    }

      // convert the input array into tf tensor
       var model_input_tensor = tf.tensor( model_input );

       model.then(function (res) {
          const prediction = res.predict(model_input_tensor).dataSync();
          var index_max = findIndexOfGreatest( prediction );
          action =  label_array[index_max] ;

       }, function (err) {
          console.log(err);
       });

       // get the predicted class
```

msgAction = action ;

Appendix 2.

Application and Data

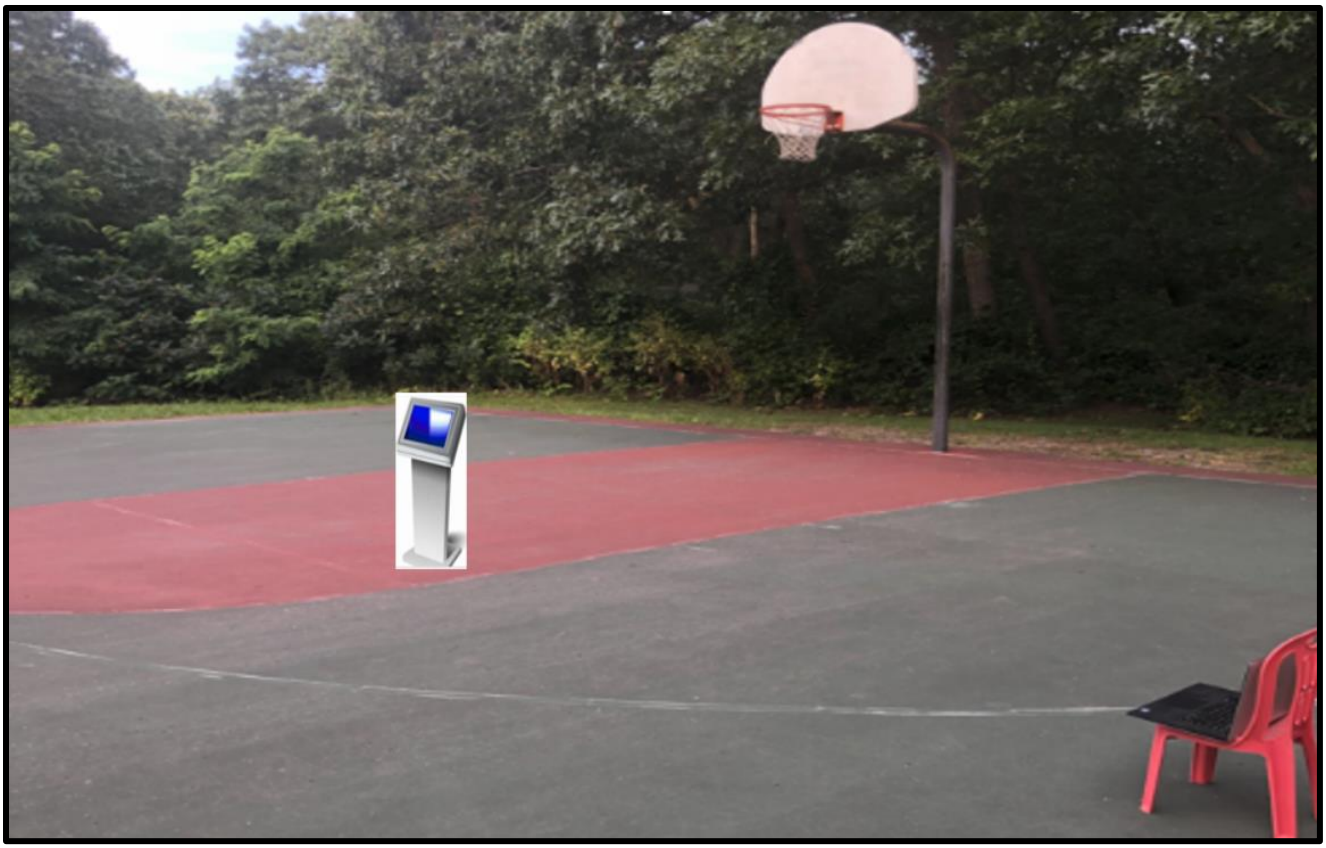
The following links are where the Application and Data reside. The Foul Shot Training Mirror is designed for a laptop, or desktop. There are URLs to view for demonstration purposes, but the design's purpose is to run locally on a client machine with a local server. This is to reduce latency and eliminate privacy concerns. See GitHub to review code and static web applications on Netlify below. See ReadMe files in GitHub repository.

1. Z-App-Xpert/PersonalFoulShotTrainingMirror_With_DNNsAndMachineLearning (github.com)
    a. This repository holds the classical programming site developed during the commonsense baseline iteration in Chapter V.
    b. '8_11_Data.csv' resides here. This data is used to train initial RNN models.
    c. Link to URL in repository. Designed to work with a modern day laptop. Full body must be in the view of the web camera.
2. Z-App-Xpert/Foul_Shot_Training_Mirror_with_GatedRecurrentUnit_GRU (github.com)
    a. This repository holds the Deep Learning Models for predicting the series, {Dribbling, Pre-Set, Set, Confidence}.
    b. Link to URL in repository. Designed to work with a modern day laptop. Full body must be in the view of the web camera.

Appendix 3.

Projection Diagram

It is an easy next step to connect an output screen closer to the shooter for more immediate feedback. The projection screen may be hard wired or wireless (Images, 2021). The shooter will have a portable monitor within their immediate sight for an immediate feedback loop to deliver the critical information to adjust their fine motor skills and "acquire, represent, and integrate knowledge." (Campero et al, 2018) Other types of projection screens will be larger and set up in basketball gymnasiums. They may be set up right behind the basketball hoop so the shooter can see their results immediately to fine-tune their motor skills immediately.



Appendix 4.

Definition of Terms

Angular Velocity - the rate of rotation around an axis usually expressed in radians or revolutions per second or per minute (Merriam Webster Dictionary, 2021).

Back Propagation – The core computer algorithm that determines the gradient descent in the most efficient manner. During back propagation, the weights and biases of the hidden layers are adjusted with the Back- Propagation algorithm (Goodfellow et al, 2016).

Classical Programming vs Machine Learning – In relation to machine learning, classical programming is taking rules and data to the program with coding structures to create answers. Machine learning takes data and answers and creates rules (Cai et al, 2020, p. 7).

Convolutional Neural Networks (CNN) - Specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers (Goodfellow et al, 2016).

Deep Neural Network – are neural networks with many layers. Modern deep learning involves many successive layers of representations of the data and are learned automatically (Cai et al, 2020, p. 13).

Edge Architecture – "Gartner defines edge computing as 'a part of a distributed computing topology in which information processing is located close to the edge—where

things and people produce or consume that information.' At its basic level, edge computing brings computation and data storage closer to the devices where it's being gathered, rather than relying on a central location that can be thousands of miles away" (Gold et al, 2021).

Gated Recurrent Unit – Gated recurrent unit (GRU) layers work using the same principle as LSTM, but they are somewhat streamlined and therefore cheaper to run. LSTM has more representational power, but GRU Neural Networks usually predict better with smaller data sets and sequences (Junyoung Chung et al, December 2014) (Chollet, Github Repository, 2018).

Gradient Descent - Is how neural networks learn. The concept of gradient descent is similar to using the chain rule of a multi-dimensional calculus problem to find the local minimum. The way the computer performs this is to assign a cost to the training data to measure the weights and biases. It starts at a random value and iteratively finds a local minimum. The study of different algorithms to find improvements and refinements on the ideas of gradient descent is ongoing (Schmidhuber, 2015), (Goodfellow et al, 2016).

HW acceleration – Term used to describe tasks being off-loaded to devices and hardware specified for the special task. Typically, machine-learning tasks are off-loaded to WebGL hardware to use GPU instead of the CPU. HW Acceleration helps MediaPipe Pose model run efficiently (Brunham, Live Perception for Mobile and Web, 2020).

Live Perception – Google's definition of any kind of machine learning that happens in the viewfinder. With the recent release of MediaPose, they use this term to describe the technology that offers the benefits of; device-local, connection-free, Privacy-conscious, Immediate, Create-in-viewfinder, Enables actions/control (Brunham, Live Perception for Mobile and Web Google Research MediaPipe, 2021).

Python Converter function – "TensorFlow.js comes with a variety of pre-trained models that are ready to use in the browser - they can be found in our models repo.

However, you may have found or authored a TensorFlow model elsewhere that you would like to use in your web application. TensorFlow.js provides a model converter for this purpose. The TensorFlow.js converter has two components:

1. A command line utility that converts Keras and TensorFlow models for use in TensorFlow.js.
2. An API for loading and executing the model in the browser with TensorFlow.js.

(Google, TensorFlow Model conversion, 2021)" (TensorFlow, 2021)

Long Short-Term Memory (LSTM) – A type of Recurrent Neural Network with feedback connections used in time series analysis (Goodfellow et al, 2016). It adds a way to carry information across many time steps. Imagine a conveyor belt running parallel to the sequence you are processing. Information for the sequence can jump onto the conveyor belt at any point, be transported to a later time step, and jump off. Intact, when you need it. It saves information for later, thus preventing older signals from gradually vanishing during processing (Chollet, Deep Learning with Python, 2018).

Recurrent Neural Networks (RNN) –are a family of neural networks for processing sequential data (Goodfellow et al, 2016). RNNS work by processing sequences of inputs one timestamp at a time and maintaining a state throughout. A state is typically a vector or a set of vectors (Cai et al, 2020).

Appendix 5.

Initial Video Trials – 12/19/2020

Name of Video	Angle	Shot Miss or Make	Observation
Video_One_ShotOne	Across from Foul Line	Miss	Writs Flick, Shoulders straight up from Hips. Wrist focused
Video_One_ShotTwo	Across from Foul Line	Miss	Shoulders straight up from hips less wrist flick, just short of hoop 3 more inches
Video_One_ShotThree	Across from Foul Line	Miss	Wrist flick Shoulders straight up
Video_One_ShotFour	Across from Foul Line	Miss	Bounced backwards after the shot
VideoOne_ShotFive	Across from Foul Line	Miss	More are wrist flip than body push
VideoTwo_ShotOne	Across from Foul Line	Miss	3 more inches. Pretty good follow through, but still leaving the hand to early because of wrist flick.
VideoTwo_ShotTwo	Across from Foul Line	Miss	Still rolling off the hand too soon, wrist flick, not released at top of momentum
VideoTwo_ShotThree	Across from Foul Line	Make	Still off wrist to soon, due to flick, but it went in after hitting the front of the rim
VideoTwo_ShotFour	Across from Foul Line	Make	Shoulders moved toward the basket and released the ball at top of momentum

Appendix 6.

Common Errors in Free Throw Shooting

The error summary below comes directly from the research of Marion Alexander (Marion Alexander et al, 2014).

1. Poor Alignment- Many shooters fail to line up the shooting side hip, knee, shoulder and elbow with a line through the ball to the basket. If any of these joints is out of alignment the shot is more likely to be released off line and miss the basket.
2. Lack of Backspin- players often apply sidespin to the ball at release; or else apply no spin at release. Both of these errors will affect the flight of the ball and may cause it to go off line en route to the basket; or to rebound off the backboard too hard or sideways and not drop into the hoop.
3. Low arc on the shot- players who do not have sufficient shoulder flexion, elbow extension or trunk extension during release often release the ball too flat; a high arc is required to ensure the ball has the maximum area of the basket to utilize on entry.
4. Relaxation of the shooting arm- the shooting arm should be completely relaxed during the shot, with only the active mover muscles contracted and all others loose and relaxed. Too much tension in the non-mover muscles of the shooting arm will interfere with the smooth release of the ball and shorten the follow-through.
5. Full follow-through after release- players should finish in the full goose neck position of the shooting hand with the arm pointing to the ceiling and the hand pointing directly to the basket.
6. Interference from non-shooting hand - If the non shooting hand is pronated or supinated at release it may move the ball out of alignment with the hoop.
7. Ball shot too hard- When a player is excited or tired they may release the ball too fast and it will bounce off the back of the rim and miss the basket.
8. Too much tension in shooting arm- shooting arm should be in full shoulder flexion, elbow extension and wrist flexion at release of the ball. If muscles are tense it may decrease the range of motion of these joints and interfere with the shot
9. Taking off at an angle- Player taking off or landing at an angle to the floor- either forward or backward- will produce an off center jump and apply non-vertical forces to the ball. Takeoff and landing should occur from the same footprints.
10. Leaning at Release- Player is either leaning forward, backwards or sideways during the release of the ball, which will produce an off center force on the ball at release.

Appendix 7.

Summary of Model Layers

In this stage, we evaluate the models and train the data to extract significant features to aid in making predictions and classifications. First, we need to read the data, scale it and scrub it. We use the sklearn library and use the function 'StandardScalar' to scale the data into a common Gaussian scale. Important parts of code are highlighted below. The full version is located here.

Vectorize Data

```
scaler = StandardScaler()
data_out = scaler.fit_transform(data_in.values.reshape(-1,1))
```

See df_scaled output as well as the simple data visualization plot to show process of scaling the data. Next, we must turn target variables into numeric values with the function LabelEncodeer

```
le = preprocessing.LabelEncoder()
```

To show a time series effect we use a vector size of 32 time events

RNN

We use simplernn layers to capture the time series effect of the input vector. With the final layer as a dense layer to map to the final target classes. We use the categorical cross entropy loss because it is a classification model.

```
conv1d --> simple rnn layers --> dense layers
model_rnn = tf.keras.models.Sequential([
 # conv 1d layer with kernel size 5
 tf.keras.layers.Conv1D(filters=128, kernel_size=5,
                        strides=1, padding="same",
                        activation="relu",
                        input_shape=[None, 6]),
 # simple rnn layers
 tf.keras.layers.SimpleRNN( 128, return_sequences=True),
```

```
  tf.keras.layers.SimpleRNN(64, return_sequences=False),
  # dense layer
  tf.keras.layers.Dense(30, activation="relu"),
  tf.keras.layers.Dense(9 , activation="softmax")
```

The LSTM is similar in design to the RNN. "It adds a way to carry information across many timestamps." (Chollet, Deep Learning with Python, 2018, p. 202) The LSTM can handle much more data to capture more complicated time series information. This data set is small, so it tends to over fit.

LSTM

```
# conv1d --> lstm layer --> dense layer
model = tf.keras.models.Sequential([
  # conv 1d layer
  tf.keras.layers.Conv1D(filters=128, kernel_size=5,
                strides=1, padding="same",
                activation="relu",
                input_shape=[None, 6]),
  # lstm layers
  tf.keras.layers.LSTM(128, return_sequences=True),
  tf.keras.layers.LSTM(64, return_sequences=False),
  #dense layer
  tf.keras.layers.Dense(30, activation="relu"),
  tf.keras.layers.Dense(9 , activation="softmax")
])
```

GRU

We have a small data set and sequenced series, so the GRU and Conv1d is the best choice due to training results. The 1D convolution layers can recognize local patterns in a sequence, which is the type of data we have (Chollet, Github Repository, 2018, p. 225). "Gated Recurrent Unit (GRU) layers work using the same principle as LSTM, but they're somewhat streamlined and thus cheaper to run" (Chollet, Github Repository, 2018, p. 215).

Since we have a smaller data set with smaller sequence, then the GRU becomes the model of choice for the Foul Shot Training Mirror.

```
model_conv1d_gru_dense = tf.keras.models.Sequential([

  # two conv1d layers
  tf.keras.layers.Conv1D(filters=128, kernel_size=3,
                         strides=1, padding="same",
                         activation="relu",
                         input_shape=[None, 6]),

  tf.keras.layers.Conv1D(filters=128, kernel_size=3,
                         strides=1, padding="same",
                         activation="relu"),
  # gru layers
  tf.keras.layers.GRU(256, return_sequences=True , reset_after=False),
  tf.keras.layers.GRU(256, return_sequences=False, reset_after=False),
  # add dropout layers to avoid overfitting
  tf.keras.layers.Dense(128, activation="relu"),
  tf.keras.layers.Dropout(0.3) ,
  tf.keras.layers.Dense(64, activation="relu"),
  tf.keras.layers.Dropout(0.5) ,
  tf.keras.layers.Dense(9 , activation="softmax")

])
```

Appendix 8.

Set-Up Instructions

The webcam must have the entire body in its view; have the selfie mode on, and pointing toward the shooting side of the person shooting the foul shot. The pictures below show the actual set-up during all development and testing. The laptop sits on the small pink chair with the webcam camera 18 inches off the ground and perpendicular to the foul line. The camera is 18 feet away from the shooter, and the specifications state the MediaPipe Pose is accurate from 19.5 feet away. In addition, there is screen-recording software running during testing to review the results. The Screen recording is currently the feedback loop.

The last photo on the left shows a mark that is on all standard basketball courts. The mark represents a perpendicular path from the foul line to the basketball hoop. The shooter uses this mark to put their shooting foot on this mark to line up their shooting stance to aim at the basket.

Figures

Figure 1. PoseNet Skeletal Diagram

Below is the Key point Diagram, sometimes referred to as the skeletal model. This model has 17 Key-points. https://github.com/tensorflow/tfjs-models/tree/master/pose-detection

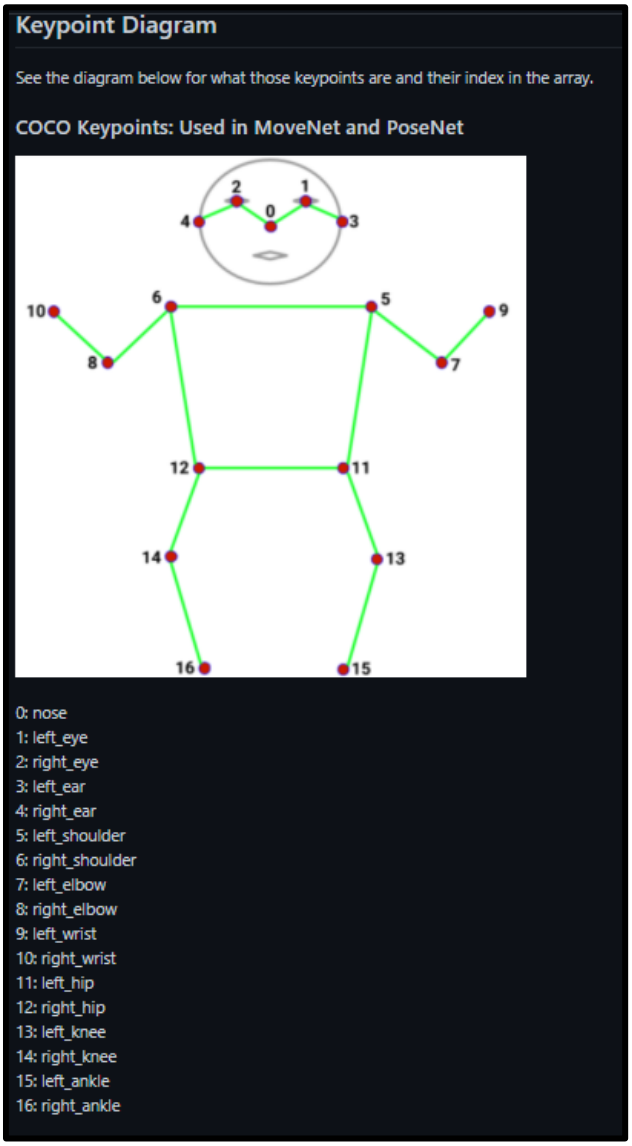
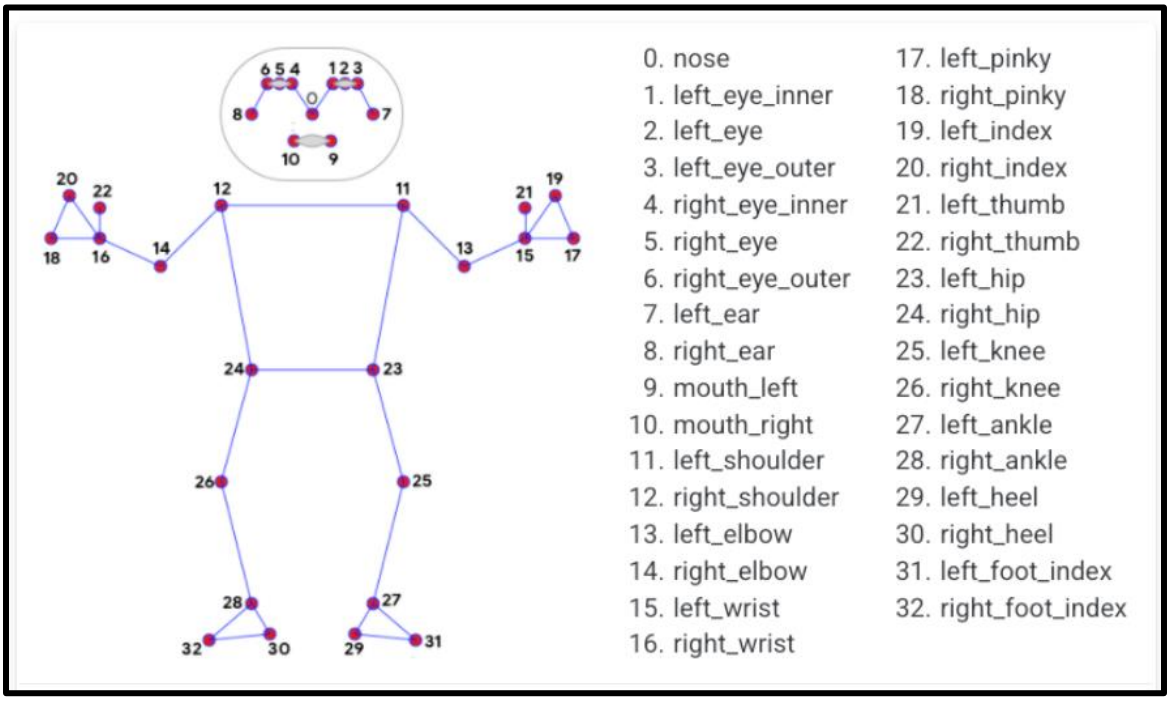


Figure 2. BlazePose Skeletal Diagram

Below is the Landmark Diagram, sometimes referred to as the skeletal model. This model has 33 pose Landmark points.

https://google.github.io/mediapipe/solutions/pose



References

Alex Krizhevsky, I. S. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM Vol.60 (6)*, 84-90.

API_TensorFlow. (2021). *TensorFlow Latest APIs*. Retrieved 2021

Brunham, M. (2020, August 24). *Live Perception for Mobile and Web*. Retrieved 07 10, 2021

Brunham, M. (2021). *Live Perception for Mobile and Web Google Research MediaPipe*. Retrieved July 1, 2021

Cai, S. (2020). *Deep Learning with JavaScript.* Shelter Island, NY: Manning Publications Co.

Campero, A. a. (2018). Logical Rule Induction and Theory Learning Using Neural Theorem Proving. *https://arxiv.org/*.

Chollet, F. (2018). *Deep Learning with Python.* Shelter Island, NY: Manning Publications Co.

Chollet, F. (2018). *Github Repository*. Retrieved 2021

Chung, J. G. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In arXiv.org (Ed.). Cornell University.

CodePen-Google. (2021). *MediaPipe - Pose*. Retrieved 2021

David J. Wright, G. W. (2018). Corticospinal excitability is facilitated by combined action observation and motor imagery of a basketball free throw. *Psychology of Sport and Exercise Volume 39*, 114-121.

Elgendy, M. (2020). *Deep Learning for Vision Systems.* Shelter Island, NY: Manning Publications CFo.

F. MartijnVerhoeven, K. M. (2016). Coordination and control of posture and ball release in basketball free-throw shooting. *Human Movement Science Vlume 49*, 216-224.

Gartner, G. (2021). *Information Technology Glossary*. Retrieved 2021

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* Sebastopol, CA: O'Reilly Media Inc.

Github_Data_File. (2021, 08 26). *PersonalFoulShotTrainingMirror_WithDNNsAndMachineLearing 8_11_Data*.

Gold, J. a. (2021, June 29). *What is edge computing and why does it matter?* Retrieved August 31, 2021

Goodfellow, I. a. (2016). *Deep Learning.* Cambridge: MIT Press.

Google. (2020). *Live ML anywhere*. Retrieved 2021

Google. (2020). *MediaPipe*. Retrieved 07 01, 2021

Google. (2020). *MediaPipe in JavaScript*. Retrieved 07 1, 2021

Google. (2020). *MediaPipe Pose*. Retrieved 7 7, 2021

Google. (2021). *TensorFlow Model conversion*. Retrieved 2021

Gruber, N. a. (2020). Are GRU Cells More Specific and. *Frontiers in artificial intelligence vol. 3, 40.*

Images, B. (2021). *Images of Computer Screen Projection Icon*. Retrieved 8 31, 2021

Ivan Grishchenko and Valentin Bazarevsky, R. E. (2020, December 20). *MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device*. Retrieved June 15, 2021

Junyoung Chung, C. G. (December 2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *NIPS 2014 Deep Learning and Representation Learning Workshop.* Cornell University.

K. He, X. Z. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

Kearney, P. E. (2017). Successful Transfer of a Motor Learning Strategy to a Novel Sport. *Perceptual and Motor Skills Volume (124(5),* 1009-1021.

Keras. (2021). *Keras API Reference*. Retrieved July 2021

Kulkarni, U. S. (2021). Quantization Friendly MobileNet (QF-MobileNet) Architecture for Vision Based Applications on Embedded Platforms. *Neural Networks, 136*, 28-39.

Lecun, Y. B. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11)*, 2278–2324.

Licklider, J. (1960). Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics, volume HFE-1*, 4-11.

Lockard, S. L. (2021). *What Is The Diameter Of A Basketball Hoop Or Rim (A To Z Guideline)*. Retrieved 06 01, 2021

Lu, Y. W. (2017). Towards unsupervised physical activity recognition using smartphone accelerometers. *Multimedia Tools and Applications*.

Marion Alexander, D. W. (2014, May 17). *Mechanics of the Basketball Free Throw*. Retrieved 06 1, 2021

Mark R. Wilson, S. J. (2009). The Influence of Anxiety on Visual Attentional Control in Basketball Free Throw Shooting. *Journal of Sports & Exercise Psychology, 31*, 152-168.

Merriam Webster Dictionary. (2021). Springfield: G. & C. Merriam Co.

NBA_TV. (2019). *Red on Roundball*. Retrieved 06 25, 2021

NPM, M. J. (2021, 7 5). *@mediapipe/pose*. Retrieved 7 5, 2021

Oved, D. (2018, May 7). *Real-time Human Pose Estimation in the Browser with TensorFlow.js*. Retrieved 05 15, 2021, from Google Creative Lab: https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html

Pujara, A. (2020, July 4). *Image Classification With MobileNet*. Retrieved 06 05, 2021

Redman, T. C. (2018, 10 11). *5 Ways Your Data Strategy Can Fail*. Retrieved 04 12, 2021

Rivera, J. D. (2020). Practical TensorFlow.js, Deep Learning in Web App Development. San Juan: Apres.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, Volume 61*, Pages 85-117.

StackOverflow. (2020). *2020 Developer Survey*. Retrieved 2021

TensorFlow. (2021). *Model conversion*. Retrieved 2021

Valentin Bazarevsky, I. G. (2020). BlazePose: On-device Real-time Body Pose tracking. *Workshop on Computer Vision for Augmented and Virtual Reality*, . Seattle, WA, USA, 2020: Cornell University.

Votel Ronny, L. N. (2021, May 17). *https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html*. Retrieved 07 01, 2021

Wong, D. W.-C.-K.-W. (2020). Effects of Upper-Limb, Lower-Limb, and Full-Body Compression Garments on Full Body Kinematics and Free-Throw Accuracy in Basketball Player. *Applied Sciences, 10(10)*, 3504.

Yang, C. L. (2021). Hierarchical Human-Like Deep Neural Networks for Abstractive Text Summarization. *IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 6*, 2744-2757.