CREATING AND COMPARING SEATED POSTURE CLASSIFICATION MODELS USING MACHINE LEARNING AND COMPUTER VISION

by

RITHIK KAPOOR

Presented to the Faculty of the Honors College of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

HONORS BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2022

ACKNOWLEDGMENTS

I would first like to start off by expressing my gratitude to Dr. Shawn Gieser for being very supportive throughout the project. He has done a great job guiding my team and making sure we have all the resources to ensure our project was a success.

I would also like to say thanks to my teammates Sri Subhash Pathuri and Dhruva Malik for pulling through all those all-nighters with me and making sure our project came out just how we envisioned it.

Finally, I would like to give my appreciation to Ms. Brown and the whole Honors College for reading through all my submissions and giving detailed feedback on them.

May 1, 2022

ABSTRACT

CREATING AND COMPARING SEATED POSTURE CLASSIFICATION MODELS USING MACHINE LEARNING AND COMPUTER VISION

Rithik Kapoor, B. S. Computer Science

The University of Texas at Arlington, 2022

Faculty Mentor: Shawn Gieser

The pandemic has led to a huge increase in the number of people teleworking these days. The growing time people spend in front of the computer elevates the risk of them developing bad posture habits that in the long run can result in health issues. It therefore becomes highly important to develop methods to monitor and correct the posture of people now adjusting to this new lifestyle of teleworking. Although there are many posture correction apps on the Google play store, almost all of them involve using some kind of expensive tracker that is placed on the body and are not widely accessible. To fix this problem, multiple posture detection models were developed to identify bad posture. We further compared the performances of these models with each other to find the model that performs better than the others.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 Background

The year 2019 resulted in a big change for people all over the world. With the rise of COVID cases, companies and governments started encouraging employees to work remotely to prevent the spread of the virus. According to an e-survey, 48% of the participants worked at home at least some time during the COVID pandemic and 33.7% worked exclusively from home. This is a huge jump from the one in twenty people that teleworked regularly in 2018 [2]. Among these employees, many of them were teleworking for the first time and did not have the experience of spending long hours in front of a screen or the knowledge of how one should maintain proper posture. It therefore becomes highly important to develop and utilize techniques to analyze posture to prevent health issues among the millions in the current generation adopting this new lifestyle. The research team decided to define good, seated posture as a posture which fulfills the following properties:

- The spine is approximately perpendicular to the thighs and is straight throughout.
- The neck is not hunched forward.
- The shoulders are not protracted forward.
- The body's weight is distributed equally and not to just one side.

On the other hand, any posture that does not follow this alignment was considered bad posture. Bad posture can result in numerous health issue such as severe low/middle back pain, moderate discomfort in eyes/neck/head, discomfort in the upper back/shoulders, and elevated stress levels [5]. Additionally, bad posture can cause long term health issues like injuries and spine curvature [6]. Identifying bad posture early can prevent a lot of these health problems that in the long term, may result in extreme discomfort and long-term damage. In recent years, a lot of progress has been made in pose estimation models that can predict various key points on the body with a high accuracy. Systems like OpenPose and BlazePose use Machine Learning and Deep Learning techniques to detect parts on the human body like eyes, shoulders, hips, etcetera [3] [4]. These models can further be leveraged to classify various poses such as good and bad posture as done in this paper. Although there are other posture correction apps on the market, almost all of them involve using some kind of expensive tracker that is placed on the body (like Upright) and do not seem to use machine learning and computer vision [7].

1.1.1 Honors Project Responsibilities

As a part of the capstone project, I worked on the additional component to create and compare machine learning models for seated posture detection. To accomplish this, I created nine models using various machine learning algorithms and by selecting different sets of key points each time. The machine learning algorithms I used were K-Nearest Neighbors (KNN), Neural Networks (NN), and XGBoost. Each machine learning model was trained on three sets of key points that were: “All Key points” that used all 33 key points, “No Limbs” that used all key points except of the limbs, and “Remove Z” that dropped the experimental z-values which represent the depth of a key point. I then tested

each of the models to find the accuracy of detecting posture, precision of detecting good posture, and precision of detecting bad posture to compare the performances of each of the models to find the best model.

1.1.2 Honors Project Tools

The tools I used while doing the Honors project were Java, Python, Tensorflow, and Mediapipe. All the machine learning models were created and tested using Python. Python has many libraries like Pandas, Numpy, Scikit Learn, and Matplotlib that improve productivity by providing commonly used functionalities needed in data science. The Tensorflow library was used to create the neural network models and convert them to TFlLite models that can run on a phone. The Mediapipe library provided a pretrained pose estimation model that allowed the extraction of 33 key points from a person in an image. Java was needed to develop our Android app and link together all the functionalities into a package that the user can easily use.

1.2 Value Proposition

Although there are other posture correction systems, almost all of them involve using some kind of expensive tracker that is placed on the body and do not seem to use machine learning and computer vision. Unlike these systems, the Machine Learning models developed as part of this capstone project only need a simple camera and can further be implemented in a variety of devices like phones, laptops, and desktops to help people maintain a good posture no matter where they go.

The models developed as part of this Capstone Project exceed in performance compared to other general models like Convolutional Neural Networks (CNNs) in terms of both accuracy and speed. This will enable people without specialized hardware to run

the model on smaller devices like phones, tablets, and laptops. Another significant advantage of the models developed in this capstone project is that they can detect good/bad posture from a variety of camera angles instead of just the front or side angle. Finally, the comparison of models created in this capstone project also provide insight into other researchers about the pros and cons of different Machine Learning algorithms when used in different scenarios.

CHAPTER 2

TECH STACK

2.1 Android Studio & Java

Initially, the research team and I were deciding whether to develop the application for android or iOS users. After analyzing multiple factors, the research team concluded that android was the better option than iOS. iOS is limited to iPhone and iPad; however Android is used as OS for a wide array of manufacturers. Unlike iOS, Android is a widely accepted platform in the world, which constitutes a major component in the market share. In addition, Android offers a lot of customization features with widgets and shortcuts while iOS only supports a few widgets. After evaluating all these key aspects, I decided to go ahead with Android OS as it will benefit more people. We had to use java as the programming language and Android Studio as framework to seamlessly see the changes on the phones. Java is a great language to use as it has multiple third-party libraries to speed up and simplify our development. In addition, Android Studio, is an integrated developer environment and an editor which makes it a one stop solution to working in the android eco system.

Further, Java is a compiled language instead of an interpreted language, like others, which makes it much faster. In this application, memory management has been taken care of automatically, which is mainly possible with the support of Java. This helps us ensure that the app does not occupy much space in the user's device.

2.2 Tensorflow

TensorFlow is an open-source library created for Python by Google Brain Team. TensorFlow compiles many different algorithms and models together, enabling the user to implement deep neural networks for use in tasks like image recognition/classification and natural language processing [8]. Object detection is a computer vision technique in which the software system can detect, locate, and trace the object from a given image or video [9]. The special attribute about object detection is that it identifies the class of object (person, part of body, chair, etc.) and its location-specific coordinates in the given image. Tensorflow detects objects by generating small segments of an image, and then feature extraction is carried out for each segmented rectangular area to predict whether the rectangle contains a valid object or not. In this way, I followed a similar approach to build on the ML Vision Kit Application to figure out the co-ordinates of the body. I have utilized the Tensorflow object detection APi to assist in constructing, training, and deploying object detection models. With this platform I could bundle together machine learning and deep learning models with a front-end design conveniently.

CHAPTER 3

METHODOLOGY

The system to detect good and bad posture in this paper is composed of two main components: the posture estimation system and the posture classification system. For each frame in a video input stream, the system uses the following steps to detect good/bad posture in that frame:

1. Detect 33 body key points using BlazePose.
2. Normalize key points by translation and scale.
3. Classify picture as good or bad posture using normalized key points via ML/DL model.

In the following subsections each one of the steps will be explained in detail.

3.1 Posture Estimation Using BlazePose

The tool used in this paper for extracting the key points from the body in the picture is BlazePose. BlazePose is a lightweight convolutional neural network architecture for human pose estimation that is tailored for real-time inference on mobile devices [3]. It detects 33 different key points on the human body as shown in Figure. 1. Figure 2 shows how it detects key points on the picture of a real human. The model consists of two main components: a pose detector and pose tracker. The pose detector is actually a fast-face detector that acts as a proxy for a person detector. The face is used as the primary feature for the pose detector because the face is the strongest signal for the neural network to predict the position of the torso. By detecting the face first, BlazePose can run at fast, real-

time speeds since it can then predict information about the pose's alignment parameters such as the middle point between hips and the size of the circle circumscribing the whole person before moving on to detecting the key points.

The pose tracker further uses the alignment parameters predicted before to detect the 33 key point coordinates in the picture and a refined area of interest to search in the next frame to increase efficiency. When compared to other pose estimation models, BlazePose was found to be 25-75 times faster on a single mid-tier phone Central Processing Unit (CPU) as compared to commonly used model OpenPose. Additionally, BlazePose can run at over 30 frames per second on a Google Pixel 2 phone [3]. Leveraging BlazePose is critical to make the posture classification system highly accessible and allow users to use it on any device that does not have specialized hardware like a laptop/mobile.

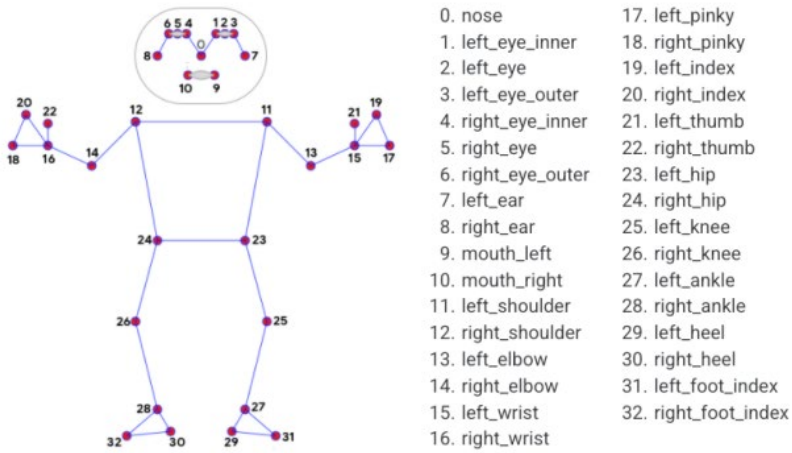


Figure 3.1: Key Points Detected by BlazePose Model

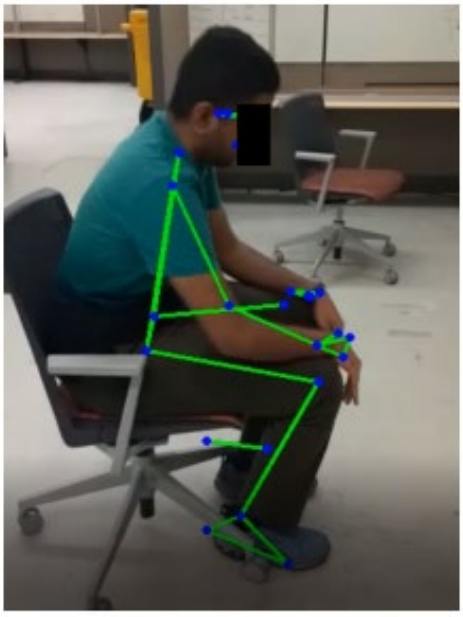


Figure 3.2: Key Points Detected by BlazePose on Real Person

3.2 Creating Datasets for Training and Testing

For creating the dataset, videos of eleven different people were taken, out of which six were used for the training dataset and five were used for the test dataset. Each participant was asked to sit in a good posture as described above and then asked to sit in their usual bad posture. A video covering a variety of angles was taken in between for each of their good/bad postures. These videos were further sliced frame by frame and analyzed by BlazePose to extract the coordinates of the key points from the frame. Next, the coordinates were normalized according to the normalization process explained below. Finally, these coordinates were saved in a CSV file to provide a better format to train ML/DL models. A total of 7,666 frames were analyzed to make up the training dataset and 5,898 frames to make up the test dataset.

3.3 Normalizing the Key Points Coordinates in the Dataset

For each frame, the translation and scale of all the key point coordinates were normalized. To normalize the translation of the pose, first the coordinates of the middle point M that lies between left hip L and right hip R were found. This gives us M = (m1 m2 m3) = 1/2(L + H) and subtract all other key point coordinates by the coordinates of M. This moved the pose center to the origin. These steps can be represented by the following matrix operations where T, X, $C \in R^{33x3}$, X is the original matrix containing the x, y coordinates of each coordinate, and T is the matrix normalized by translation.

$$T = X - C$$

$$C_{33\times3} = \begin{pmatrix} m_1 & m_2 & m_3 \\ m_1 & m_2 & m_3 \\ \vdots & \vdots & \vdots \\ m_1 & m_2 & m_3 \end{pmatrix}$$

s

Figure 3.3: Formula for Normalizing the Translation of Poses

Finally, to normalize the scale of the pose, first, the maximum distance between any key point to the pose center M is calculated. This distance is considered as the scaling factor for that particular human subject. Hence, all the body key point coordinates are divided by the calculated maximum distance which normalizes the scale of each pose resulting in all coordinate values between 0 to 1. The above steps can be represented by the following matrix operations:

$$S_{33\times1} = \begin{pmatrix} ||t_1|| \\ ||t_2|| \\ \vdots \\ ||t_{33}|| \end{pmatrix}, \text{where } T = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_{33} \end{pmatrix} \text{and } ||t|| = \sqrt{\sum_{i=1}^{3} |t_i|^2} \text{ (L2-norm of } t)$$

$$N_{33\times3} = \frac{T}{Max(S)}$$

Figure 3.4: Formula for Normalizing the Scale of Posture

Here, T is the matrix normalized by translation and N is the final matrix normalized with respect to both translation and scale.

3.4 Training the Machine Learning Models

From the first phase, all body key points extracted from the image frames using BlazePose were normalized and stored in separate CSV files for each subject. The CSV files containing normalized data were only for training and testing purposes. Once the model was deployed, the extracted key points were classified in real-time and did not need to be stored anywhere. The pre-processed normalized body key points were used to train and compare three different sets of ML models: K-Nearest Neighbors (KNN), XGBoost, and Neural Networks (NN). For the KNN model, I used the 10 nearest neighbors with uniform weights for classification. For the XGBoost model, I used the tree booster with an eta (learning rate) of 0.3, and max depth of 6. Finally, for the Neural Network model, I used three hidden layers with 100 perceptrons each, a learning rate of 0.001, Relu as the activation function, Adam as the optimizer, and trained the model over 100 epochs.

In addition to using different models, the combination of key points used for training was also varied. For the first case, all 33 body key points (All Key points) were used for training. Next, the models were trained excluding the key points from the limbs

(No Limbs). Finally, the experimental depth values (z-values) were dropped and only x and y coordinates were used from all 33 key points to train the third set of models (Remove Z).

CHAPTER 4

RESULTS

The results from each set of training for the three models can be found in Table 1.

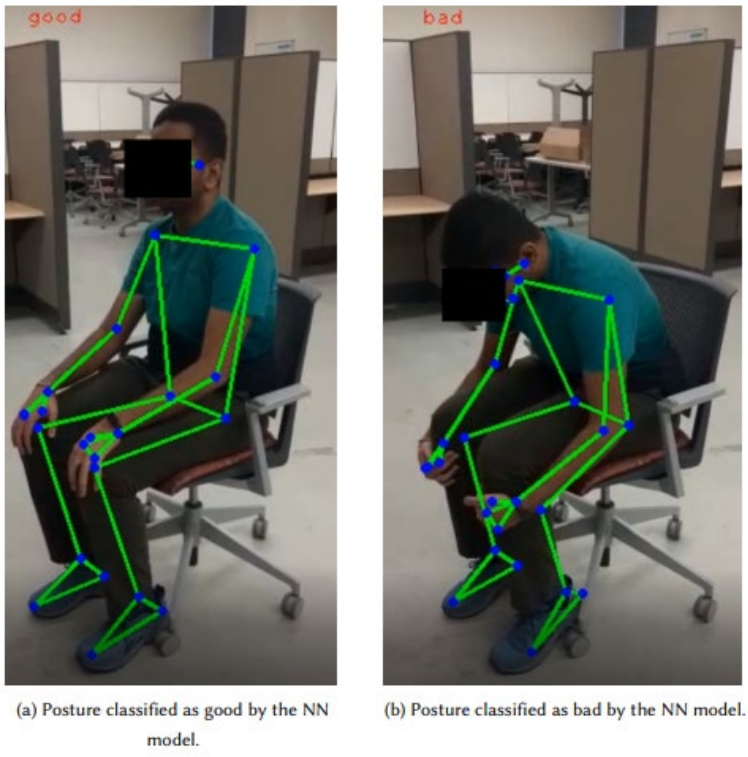
Table 1. Performance comparison among the different models: **XGBoost**, **Neural Networks (NN)**, and **K-Nearest Neighbors (KNN)**. Acc. represents the accuracy of each model on three different combination set of keypoints: All Keypoints, No Limbs, and Removed-Z. P_G and P_B represent the precisions for classification of Good and Bad postures respectively.

Keypoints	XGBoost			NN			KNN		
	Acc.	P_G	P_B	Acc.	P_G	P_B	Acc.	P_G	P_B
All Keypoints	0.92	0.87	0.98	**0.98**	0.97	0.99	0.92	0.87	0.98
No Limbs	0.89	0.82	0.98	0.93	0.88	0.98	0.92	0.86	0.99
Removed-Z	0.92	0.87	0.99	0.94	0.92	0.96	0.96	0.92	0.99

Table 4.1: Results of Different Module Comparison

The best accuracy and precision were achieved with the Neural Network model using all key points at 98% accuracy. I also observed that I got a decent accuracy of 93% with the no limbs Neural Network model. This means that the posture detection algorithm could be sped up by just detecting 15 key points instead of the original 33. Finally, it was observed that removing the Z features does not have too much of an effect on the performance of the models and was in fact performing better in the case of the KNN model. This suggests that adding a depth sensor would not have benefited the models much and an (Red-Green-Blue) RGB camera should suffice for our purposes. All the above models appear to maintain a constant real time performance above 20 Frames-Per-Second (FPS) without any drop in accuracy on an i7-8750H laptop.

The KNN model was further integrated in an Android app and achieved a real time performance of 7-9 FPS on a Samsung A10. On a laptop with an i7-8750H, the model's performance improved to 25-28 FPS. This demonstrates that the posture system explained in this paper is accessible on everyday devices and does not require specialized hardware like Graphics-Processing-Units (GPUs). Example of the system classifying good and bad posture can been seen in Figure 4.2.



(a) Posture classified as good by the NN model.

(b) Posture classified as bad by the NN model.

Figure 4.1: NN Model Classification Results

CHAPTER 5

CONCLUSION

I created nine models using various machine learning algorithms and by selecting different sets of key points each time. The machine learning algorithms used were K-Nearest Neighbors (KNN), Neural Networks (NN), and XGBoost. For each machine learning model, the three sets of key points they were trained on were “All Key Points” that used all 33 key points, “No Limbs” that used all key points except of the limbs, and “Remove Z” that dropped the experimental z-values which represent the depth of a key point. I then tested each of the models and found that the Neural Network model using all 33 key points performed the best with 98% accuracy. With people spending most of their time in front of a computer, it becomes highly important to maintain proper posture to avoid any long-term health issues. This is where the models come into play being fast, efficient, and able to provide real-time feedback. Creating these models for my capstone project further allowed me to find the best possible models that my team and I could use for our senior design project and really enhanced our project with its great performance.

REFERENCES

[1]: Cao, Zhe, et al. "Realtime multi-person 2d pose estimation using part affinity fields." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[2]: European Foundation for the Improvement of Living and Working Conditions Page 41. "Living, Working and COVID-19—First Findings—April 2020." (2020).

[3]: Bazarevsky, Valentin, et al. "Blazepose: On-device real-time body pose tracking." *arXiv preprint arXiv:2006.10204* (2020).

[4]: Osokin, Daniil. "Real-time 2d multi-person pose estimation on cpu: Lightweight openpose." *arXiv preprint arXiv:1811.12004* (2018).

[5]: AlOmar, Reem S., et al. "Musculoskeletal symptoms and their associated risk factors among Saudi office workers: a cross-sectional study." *BMC Musculoskeletal Disorders* 22.1 (2021): 1-9.

[6] : Gerding, Thomas, et al. "An assessment of ergonomic issues in the home offices of university employees sent home due to the COVID-19 pandemic." *Work* Preprint (2021): 1-12.

[7]: "Everyday Posture Coaching." *UPRIGHT Posture Training Device*, 12 May 2022, https://www.uprightpose.com/?gclid=CjwKCAjw4ayUBhA4EiwATWyBrh2wvpzrfq4dhVQlkObwxyuZM1sKuPs0PFrinlTLIo0HYqkieiit_hoC2C4QAvD_BwE.

[8]: "Why Tensorflow." *TensorFlow*, https://www.tensorflow.org/about.

[9]: “ELI5: What Is Image Classification in Deep Learning?” *ThinkAutomation*, 3 Dec. 2020, https://www.thinkautomation.com/eli5/eli5-what-is-image-classification-in-deep-learning/.

BIOGRAPHICAL INFORMATION

Rithik Kapoor is a third-year undergraduate student studying computer science at the University of Texas at Arlington. He works as a research assistant under Dr. Fillia Makedon in the Heracleia Lab. His prior experience includes working as a data science intern at Axtria, where he worked on the automation of ML workflows. His interests primarily lie in the fields of machine learning and computer vision.