## Inhaltsverzeichnis

## Software idea

The current Covid-19 crisis has forced students to work at their home desks, which may lead to back pain and other health issues associated with a poor sitting posture. Thus, the software described below serves to analyse the sitting posture of human bodies and offers recommendations based on the results of the analysis. Thereby, the analysis focuses on two common postural errors: forward head posture and lateral pelvic tilt.

## Coding environment

The back end was coded in python in Google Colab and we were using the accessible GPU. Also, we were able to use the applied version of OpenPose in the Google Colab environment and would have had to use another version or tweak it to work on our local machine. Furthermore, through the combination of Colab and Github we were able to work jointly on the projects and review the code in our shared Github repository. Also, each of us had our own branch in Github for the various parts where we pushed our codes to regularly.

The web application was developed using HTML and CSS and implemented through research, bootstrap and further webdesign resources. The decision to go towards a web application instead of a desktop or mobile application was based on the fact that a web app will allow for easier accessibility because it can be accessed through the internet. Also, this way users do not need to download and install an application on their device. In the long-term we plan to find a more scalable solution to this. Furthermore, developing a web application led us to delve into HTML and CSS languages which will also prove a useful learning experience for future projects. The various HTML implementations include for example elements such as buttons, video tags, video placeholders, headlines and further features visible in the web application.

For the communication between the homepage and the code we use Flask since it works well with ngrok and can be implemented inside Google Colab, too. Flask is a python framework for building web applications and creating the communication between client and server. In particular, it can be described as a microframework which increases the speed of deployment and testing. Flask depends on the Jinja template engine and the Werkzeug WSGI toolkit libraries. Ngrok is a cross-platform application that serves as a mediator between a local server and the internet. Google Colab provides a virtual machine, so we cannot access the localhost (all it does it route it to our local machine's localhost) as we do on our local machine when running a local web server. What we can do is expose it to a public URL using ngrok. Here comes the Python library flask-ngrok into play. However, this server is only a development server which is why it shouldn't be used in production environments. In the long-term we plan to find a more scalable solution to this.

As input for the model as well as for the homepage we chose to use video files instead of images because these hold various advantages, and our model is able to work on video files, too. The advantages include better monitoring for statistics and present a more realistic use case where posture is tracked over a longer period instead of just a singular snapshot.

Nevertheless, the applied model works with "images", too. This happens by segmenting the video into various frames and analysing each frame. The output are .json files which we can then analyse in python by importing the json library. In these files the x and y coordinates of all the various recognised keypoints are stored. In our model we are looking particularly at the left and right shoulder which refer to the keypoints 2 and 5 (s. fig 1)[1].

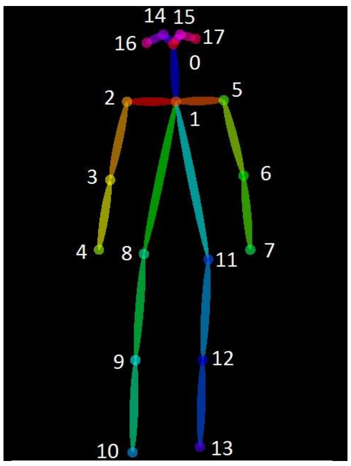

*Figure 1: OpenPose 18 keypoints*

The model also delivers a video output with the submitted video as source but being enhanced through highlighting all the identified keypoints.

In the next step we retrieved the various x and y coordinates of the relevant keypoints from the .json files by performing manipulations on dictionaries and lists. We stored these values so we could access them later and perform operations on them. We were doing this by performing various calculations such as distance and slope and generated new values from this. These new values were then smoothed to avoid outliers by being averaged with their 4 closest neighbours in the list.

We used the matplotlip library to then plot the smoothed distance and slope values to visualize where the recognised postures deviate from the should-be case. These graphs were then stored as images which are ready to be pushed back to the website.

The push back to the website includes four elements: the submitted video enhanced through the identified keypoints, the two plotted graphs, and the recommendations which we send back to the user. For this we use the post method sending HTML from data to the server. The data received by the POST method is not cached by the server.

Last but not least, we also imported os and glob libraries. OS library provides a portable way of using operating system dependent functionality. Glob module finds all the pathnames matching a specified pattern which is very important in communication with the drive and the web application. Furthermore, we mounted our Google Drive to the Colab, so we can quickly access our files which was very important in the first testing and development phases.

## Workflow from user perspective

Users can easily access the web application and find a range of buttons there. Hierarchically the highest button allows the user to upload a video as .mp4 file by clicking the "submit" button. This video will be uploaded, and the user can see a confirmation in the headline of the webpage.

When the user clicks on the button "analysis" the analysis is started in the background. After a short moment the user will see the analysed version of his video which includes a highlighting with all the identified relevant keypoints. Meanwhile, he also receives two .png

[1] Kanan, Amer. (2018). Implementation of Agriculture Information Management (AIM) in Palestine, p.66.

files as feedback which are the described graphs for the shoulder distance and the slope changes in the shoulders. These values closely relate to certain diseases and the user can see the deviations of the measured values to the should-be range of values as indicated by a red line in the graphs. Furthermore, the user receives as feedback specific recommendations regarding prevention, education and treatment with regards to the diagnosis.

## The underlying model - OpenPose

As basis of our project served the OpenPose model[2] which allows for multi-Person 2D Pose Estimation using part affinity fields (PAF). By combining PAFs with body part confidence maps, the model allowed us avoiding common pitfalls in pose estimation. Upon this combination, the maps are parsed using Bipartite Matching to associate body part candidates.

When taking a closer look at the underlying Convolutional Neural Network, we can see that for each stage, the predictions and their corresponding image features are concatenated to the subsequent stages (s. fig 2). Thus, the model iteratively predicts affinity fields that encode part-to-part association and 2D detection confidence maps. With this approach the prediction ability increases over the various stages while each stage is directly supervised. PAFs are crucial for assigning a body part specifically to one person and therefore enables multi-person estimation. They preserve both location and orientation information across the region of the limb. Each PAF can be expressed as a 2D vector field for the various body parts.

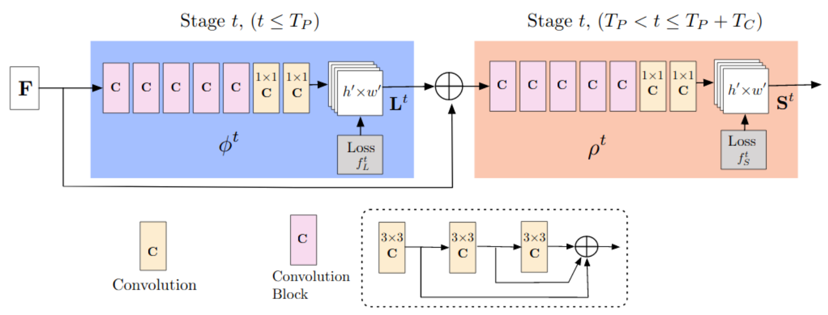

*Figure 2: Architecture of the multi-stage OpenPose model*

[2] Cao, Z., Hidalgo, G., Simon, T., Wei, S., Sheikh, Y. (2018): OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In: arXiv e-prints. https://ui.adsabs.harvard.edu/abs/2018arXiv181208008C.