T431 – Applied A.I. Solutions Development

Human Posture Detection

Prepared for

FULL STACK DATA SCI. SYSTEMS COURSE

Prepared by:

BESJANA BEQIRI

CHHAYANK

INDERPREET SINGH VOHRA

JAGATHI POOVARAGHAVAN

PRATEIK MALLA

Delivered on:

August 19, 2022

Table of Contents

Introduction 3
1. Problem Statement 3
2. About Dataset 3
   2.1. Data preprocessing 3
3. Methodology 4
   3.1. Competition 5
4. Deployment 5
   4.1. Challenges 7
5. Conclusion 8
   5.1. Demo 9
Reference 10

Introduction

Our company POSCO, is developing a product called Personal Posture Coach, to help employees of all ages improve their posture whenever they are sitting in front of a machine with a webcam on it, sending them a pop up for correcting their posture. The POSCO mission is to improve the posture of the people in their workplace, but not only, by being their personal coach, whenever they need it.

We are a leader in Cloud based Informatics solutions. With our deep laboratory domain expertise, we provide software development and integration services for implementing our solutions, your solutions, or our combined solutions. Our teams solve some of the biggest information technology challenges facing laboratory businesses today.

1. Problem Statement

Many working-class people are facing various medical conditions directly related to sedentary lifestyles. One of the frequently mentioned problems is back pain, with bad sitting posture being one of the compounding factors to this problem. Poor posture contributes to a variety of health issues, including headaches, eyes fatigue, and discomfort to your body, especially legs, wrist, shoulder, neck and back. Clearly, having a mechanism or product which helps in avoiding sedentary lifestyle with poor sitting habits can be a project with huge market opportunity.

In order to respond to management's call for rapid introduction of more successful new products, the technical team has come up with an Artificial Intelligence product that can be embedded into a laptop as a software application or web interface which uses image processing techniques to detect and alert the users sitting posture.

2. About Dataset

We created our own dataset, by taking pictures from the webcam. We took 70 good posture pictures and put them in a folder. We took 125 bad posture pictures and put them in another folder. So, we have two folders with good and bad posture samples.

2.1. Data preprocessing

Image pre-processing are the steps taken to format images before they are used by model training and inference. Pre-processing is required to clean image data for model input. For example, fully connected layers in convolutional neural networks require that all images are the

same sized arrays. Image pre-processing may also decrease model training time and increase model inference speed. If input images are particularly large, reducing the size of these images will dramatically improve model training time without significantly reducing model performance. In our case we made sure about the Image size, aspect ratio, format, RGB color formatting. Using the models, we build landmarks which can be used to predict the position during training.

3. Methodology

In order to implement a computer vision project, we will require a strong technical team skilled in computer vision analyst, data architects, data management resources, solution architects, software developers, web developers along with infrastructure to build advanced machine learning products that are cloud services for storage, processing, deployment, hosting etc. And most importantly, a research team that can study healthy sitting postures and convert the medical information into quantifiable metrics for defining the correct and incorrect image postures for the deep learning models to be trained with better accuracy. There are a few approaches towards packaging a solution for this problem requirement which can either use 3D video streaming posture recognition that involves live video streaming input collected from camera setup from different angles to record the postures of human beings and utilize the video analysis techniques to classify the postures as correct or incorrect and live stream the video into a software application along with alerts whenever there is incorrect postures detected, another approach is to use images from live-capture which can again be from different angles or front facing from web camera to record the postures and use image detection and classification algorithms to repeat the same process. The latter process can be cost effective since it involves image files rather than video files that reduces storage and model training cost and can still provide close accuracy compared to video-captured analysis.

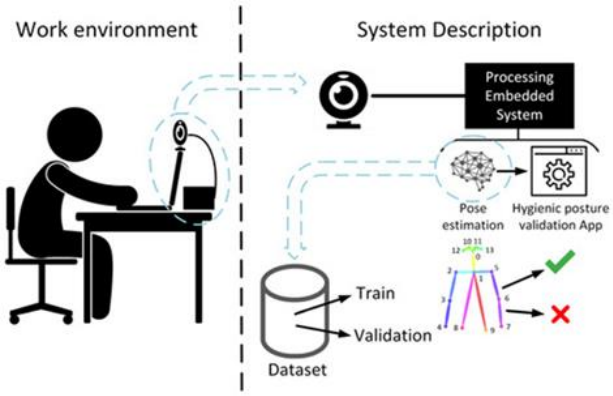

Figure 1. High Level Design.

The input images are captured from the laptop camera or web camera. When the application is opened or set in background the camera captures the person's posture looking at the laptop and the machine learning model which has the parameters processed and defined for posture will classify the image as correct or incorrect posture based on the training. So, the end result will be an alert or message displayed with your image that depicts whether your posture is correct or not.

To train our model we needed to choose between the most popular pose estimation models: PoseNet, MoveNet, Openpose. After doing a few experiments with the 3 models we went for MoveNet, as it was faster for our application. To build our web app, as a requirement for the frontend part we used TFlite files embedded in javascript , where TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices. We needed to decide on the cloud platform for storage & web app hosting. So, we chose google cloud, because it was easy to use, less access restrictions as we have accounts, and had exposure to using google tools already.

3.1. Competition

When we try to study the competition for our product while deciding marketing and sales strategies. We noticed the following players. Primarily there are a lot of free google chrome extensions which help in posture recognition, but all of these are timed by the users themselves and the extension simply nudges for stretching, drinking water, and more throughout the day without knowing the real-time posture. Although they don't directly help in posture correction, since it is available for free in the market it can be a competitor for our product, hence we chose to make it an enterprise app rather than targeting customers worldwide.

Apart from this, we have other devices which do not involve A.I. products but serves the same problem statement and can be a competitor such as smart chairs with IoT sensors and Yoga straps with posture correctors using mobile apps to track the posture.

And the A.I Product competitor will be 3D video posture recognition which involves extra infrastructure setup to capture the 3D video stream of human sitting.

The last three products involve continuous investment from the customer to setup and purchase devices, apps and then pay for subscription too while our product will solve the same problem with minimal investment required from development and customer standpoint. Also, compared to free web extensions we use real-time images so our product will have better accuracy in solution.

4. Deployment

MoveNet is a high-speed position tracker. The model is pre-trained, so it is ready to use after setting up. It tracks ankles, knees, hips, shoulders, elbows, wrists, ears, eyes, and nose, for a total of 17 key points. MoveNet versions: Lightning and Thunder. Lightning is faster but may produce less accurate results. Thunder is slightly slower, but more accurate. According to TensorFlow, both can run at 30+ FPS. MoveNet Detect & Preprocessor to create landmarks and build csv files. Torso Multiplier increased because we had an image only until chest or hip. Capture Coordinates and Center Point score for Eyes, Ear, Shoulder, Hip, Train & Test Split 80:20. Once trained, we convert model to tflite **TENSORFLOW Lite:** is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices

TensorFlow Lite is available in the Google services API. The API lets you run machine learning (ML) models without statically bundling TensorFlow Lite libraries into your app, allowing you to: Reduce your app size.

```
[23] converter = tf.lite.TFLiteConverter.from_keras_model(model)
     converter.optimizations = [tf.lite.Optimize.DEFAULT]
     tflite_model = converter.convert()

     print('Model size: %dKB' % (len(tflite_model) / 1024))

     with open('pose_classifier.tflite', 'wb') as f:
       f.write(tflite_model)

     with open('pose_labels.txt', 'w') as f:
       f.write('\n'.join(class_names))
```

```
Classification Report:
              precision    recall  f1-score   support

         Bad       0.90      1.00      0.95         9
        Good       1.00      0.80      0.89         5

    accuracy                           0.93        14
   macro avg       0.95      0.90      0.92        14
weighted avg       0.94      0.93      0.93        14
```

There are four ways to check if the predictions are right or wrong [1]:

TN / True Negative: the case was negative and predicted negative
TP / True Positive: the case was positive and predicted positive
FN / False Negative: the case was positive but predicted negative
FP / False Positive: the case was negative but predicted positive

Precision — *What percent of your predictions were correct?*

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive.
Precision: - Accuracy of positive predictions.
Precision = TP/ (TP + FP). In our case, for the class Bad Posture the Precision is 0.9 and for the class Good Posture is 1.
Recall — *What percent of the positive cases did you catch?*
Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.
Recall: - Fraction of positives that were correctly identified.
Recall = TP/ (TP+FN). In our case, for the class Bad Posture the Fraction of positives that were correctly identified is 1 and for the class Good Posture is 0.8.

F1 score — *What percent of positive predictions were correct?*
The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.
F1 Score = 2*(Recall * Precision) / (Recall + Precision). In our case, for the class Bad Posture the F1 score is 0.95 and for the class Good Posture is 0.89.

Support
Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

4.1. Challenges

There were numerous challenges that we faced while doing this project. The test data set of the images had a bias in it as it was only collected from one single person. We needed to work on the bias by collecting posture data from multiple people. The center distance of the body is different for every individual and to calculate it for each person we needed to train our model a lot more as it assigns the distance a probability score which is used to detect whether a person is sitting correctly or not. We need to work on that. Since we focused more on the bad posture in order for our model to detect it correctly, we need to introduce our model with correct postures as well in order to obtain correct F1 score.

5. Conclusion

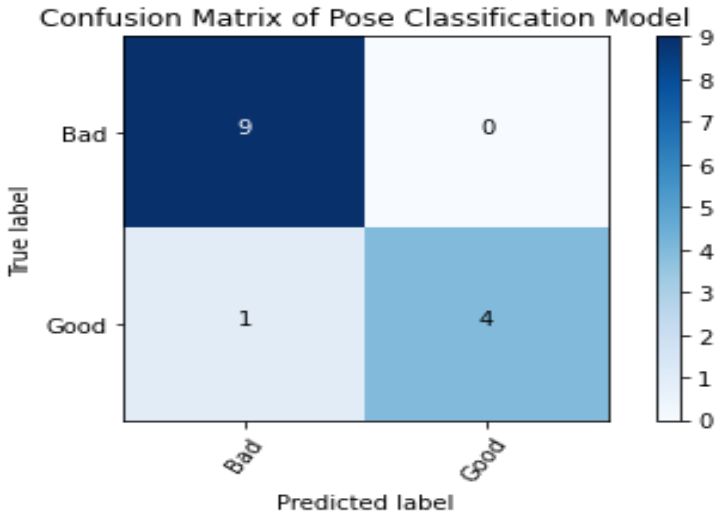

Figure 2. The confusion matrix for the model.

A confusion matrix [2] is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm. A confusion matrix is shown in Figure 2, where *Bad* means having a bad posture and *Good* having a good posture.

The confusion matrix consists of four basic characteristics (numbers) that are used to define the measurement metrics of the classifier. These four numbers are:

1. TP (True Positive): TP represents the number of users who have been properly classified in Bad class, meaning they have a bad posture. In our case this number is 9.
2. TN (True Negative): TN represents the number of correctly classified users who are having a Good posture. In our case this number is 4.
3. FP (False Positive): FP represents the number of misclassified users with the Bad posture but actually they are having a Good Posture. In our case this number is 1.
4. FN (False Negative): FN represents the number of users misclassified as having a God posture, but actually they are having a Bad Posture. In our case this number is 0.

As we can notice our model is a little biased and it predict better the Bad postures than the good ones.

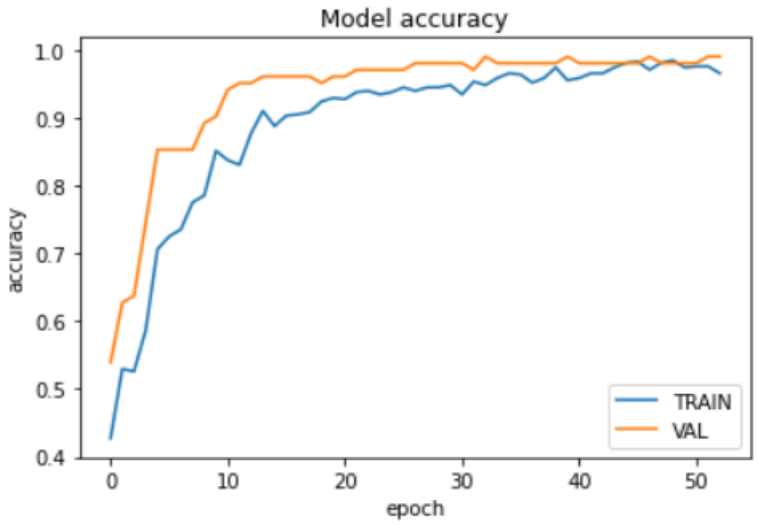

Figure 3. Training and Validation accuracy.

Results

- Test Accuracy: 94%
- Validation Accuracy: 92%

In the Figure 3, we have an Overall Accuracy range between 92-95% and Overall Loss range between 0.19 - 0.12.

5.1. Demo

https://posturecoach-3171b.web.app/

Reference

[1] https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397

[2] https://www.sciencedirect.com/topics/engineering/confusion-matrix#:~:text=A%20confusion%20matrix%20is%20a,malignant%20tissue%20is%20considered%20cancerous.