POSCO Your Personal **Po**sture **Co**ach

A project for

FULL STACK DATA SCIENCE SYSTEM

By

Group :- 1

Besjana Beqiri

Chhayank

Inderpreet Singh Vohra

Jagathi Poovaraghavan

Prateik Malla

Agenda

01

INTRODUCTION

02

DATA PREPROCESSING

Creating dataset and preparing it for training models

03

DL MODEL & FRONTEND

Building pose estimation model
Train and test the model
Wrap it as an web app using html or javascript

04

DEPLOYMENT

Deploy web app using Google tools

05

CONCLUSION & DEMO

Conclusion and live demonstration of the product

Problem Statement ?

According to the Arthritis Disease Center, 50- 80 % of people complain of back pain in the workplace, because of Bad posture.

2015

Michael started to work from home.

2018

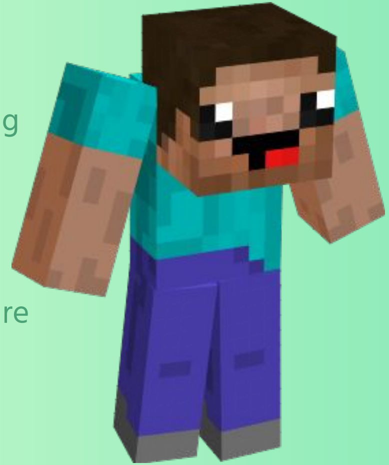
Bad sitting posture ,Slouching while working from home

2019

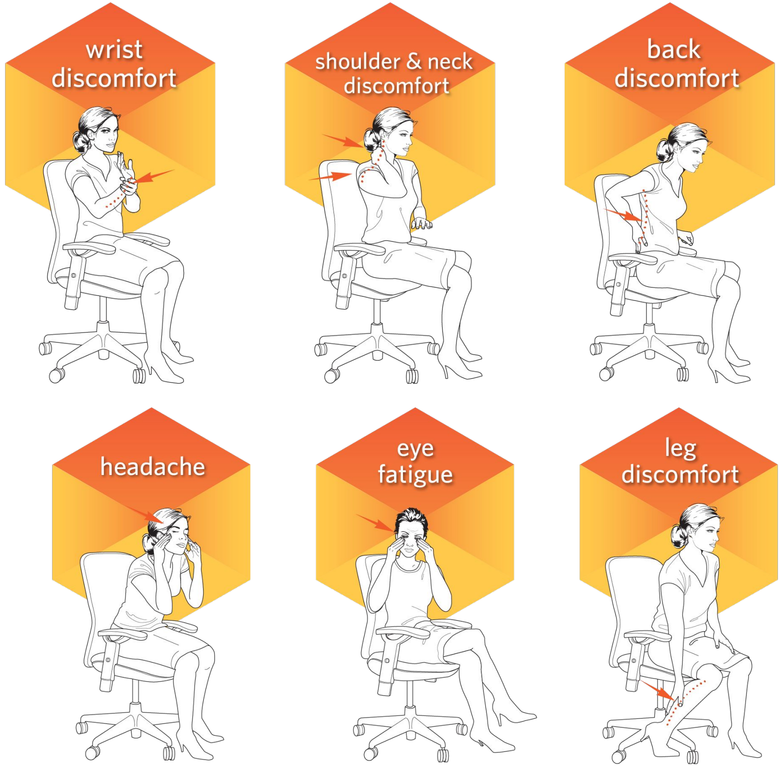
Due to lack of any posture detection software the person developed permanent Hunchback

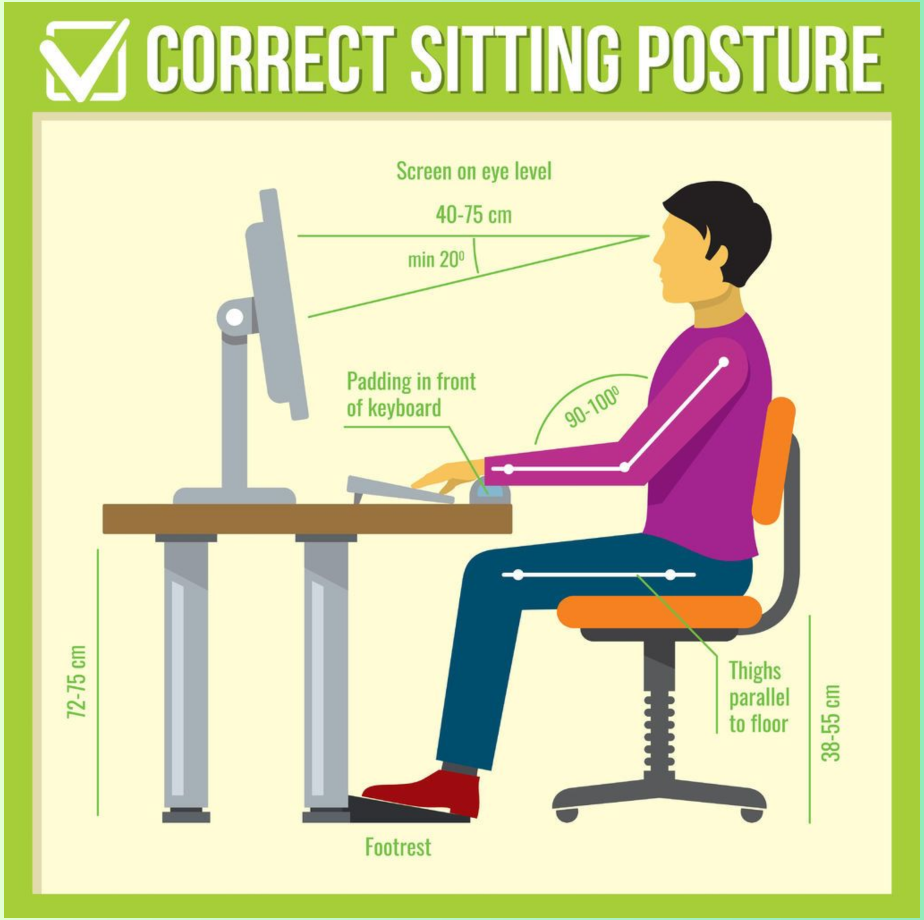
❑ Age: 25-35

❑ Gender: male



What is a correct posture…?
What happens if we don’t have a correct posture…?





DATA CAPTURE & PREPROCESSING

CREATING DATASET

70 Good Postures in one folder
125 Bad Postures in one folder

PREPROCESSING

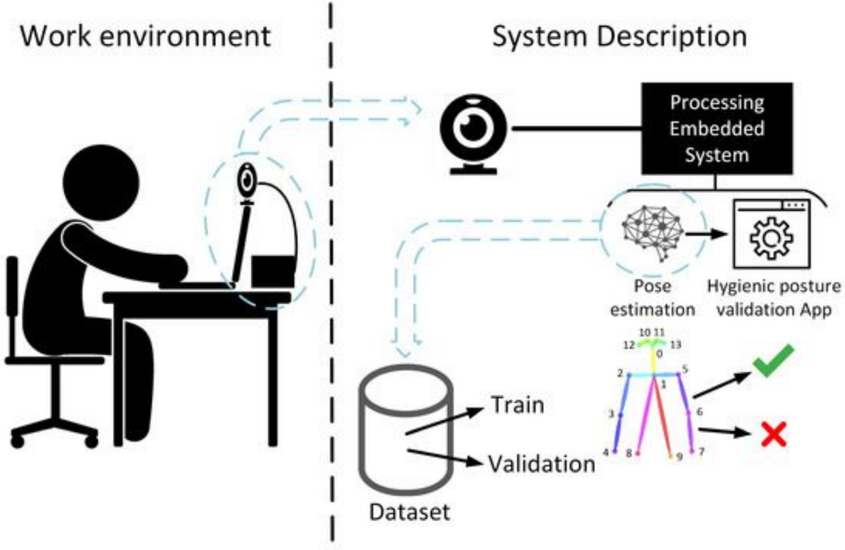
Using the models we build landmarks which can be used to predict the position during training.

STORING

We stored it as a csv file.

Proposed Methodology

The Idea



- Using the webcam for image collection and the final web app.

- Decide on the Popular pose estimation models: PoseNet, MoveNet, Openpose.

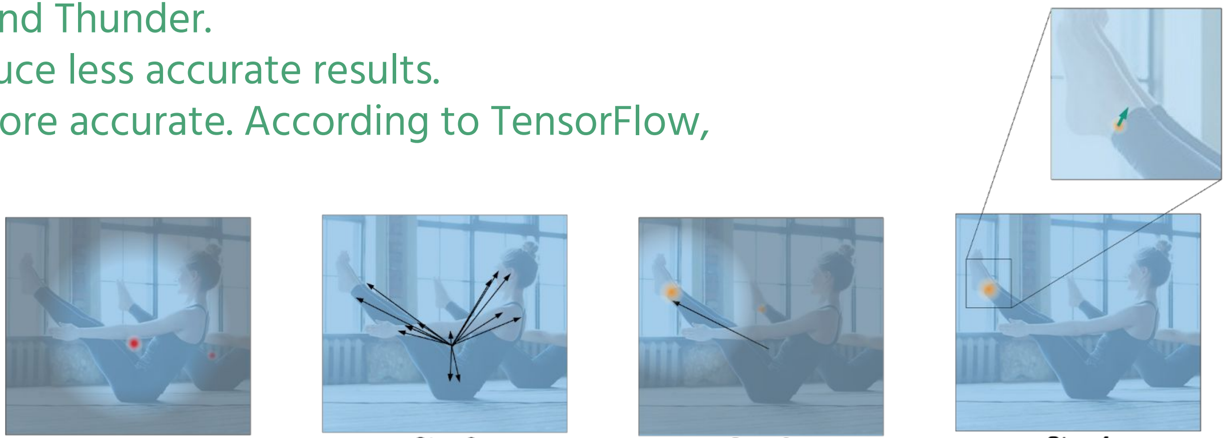
- Decide on the Frontend requirements, with MoveNet we will have TFlite files embedded in javascript.

- Decide on the cloud platform for storage & web app hosting.
  We chose google - easy to use, less access restrictions as we have accounts, had exposure to using google tools already.

MoveNet

MoveNet is a high-speed position tracker. The model is pre-trained, so it it is ready to use after set up! It tracks ankles, knees, hips, shoulders, elbows, wrists, ears, eyes, and nose, for a total of 17 key points.

MoveNet versions: Lightening 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.



Step 1
Weight the object center heatmap based on the inverse distance from frame center. Compute the location of the maximum heatmap value.

Step 2
Slice out the keypoint regression vector at the peak center location.

Step 3
Weight each keypoint heatmap based on the inverse distance from the regressed location. This attenuates the scores from the background keypoints.

Step 4
Compute the location of the maximum heatmap value, and add the local 2D offset at that location.

Fig: Movenet PreProcessing

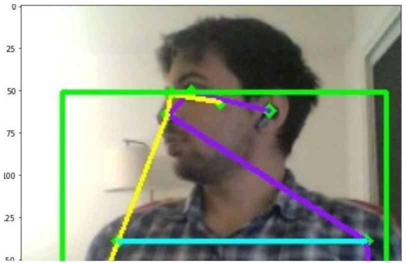
MoveNet

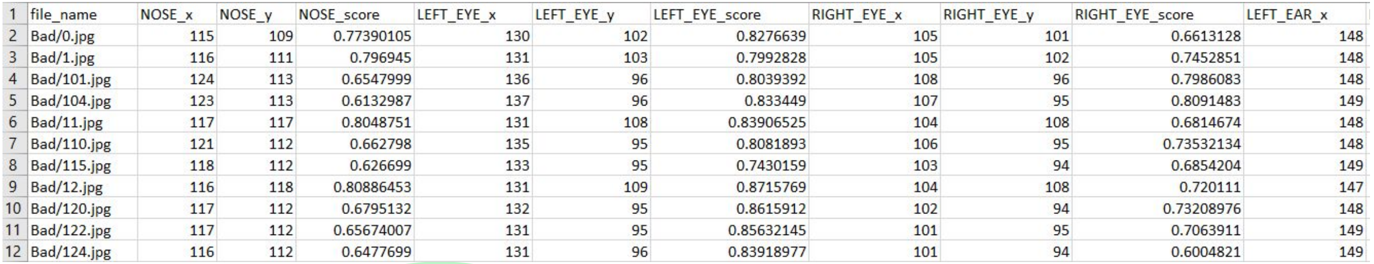
MoveNet Detect & Preprocessor to create landmarks and build csv files

Torso Multiplier increased because we had image only until chest or hip

Capture Coordinates and Center Point score for Eyes, Ear, Shoulder, Hip

Train & Test Split 80:20





1	file_name	NOSE_x	NOSE_y	NOSE_score	LEFT_EYE_x	LEFT_EYE_y	LEFT_EYE_score	RIGHT_EYE_x	RIGHT_EYE_y	RIGHT_EYE_score	LEFT_EAR_x
2	Bad/0.jpg	115	109	0.77390105	130	102	0.8276639	105	101	0.6613128	148
3	Bad/1.jpg	116	111	0.796945	131	103	0.7992828	105	102	0.7452851	148
4	Bad/101.jpg	124	113	0.6547999	136	96	0.8039392	108	96	0.7986083	148
5	Bad/104.jpg	123	113	0.6132987	137	96	0.833449	107	95	0.8091483	149
6	Bad/11.jpg	117	117	0.8048751	131	108	0.83906525	104	108	0.6814674	148
7	Bad/110.jpg	121	112	0.662798	135	95	0.8081893	106	95	0.73532134	148
8	Bad/115.jpg	118	112	0.626699	133	95	0.7430159	103	94	0.6854204	149
9	Bad/12.jpg	116	118	0.80886453	131	109	0.8715769	104	108	0.720111	147
10	Bad/120.jpg	117	112	0.6795132	132	95	0.8615912	102	94	0.73208976	148
11	Bad/122.jpg	117	112	0.65674007	131	95	0.85632145	101	95	0.7063911	149
12	Bad/124.jpg	116	112	0.6477699	131	96	0.83918977	101	94	0.6004821	149

MoveNet

Batch Size: 16

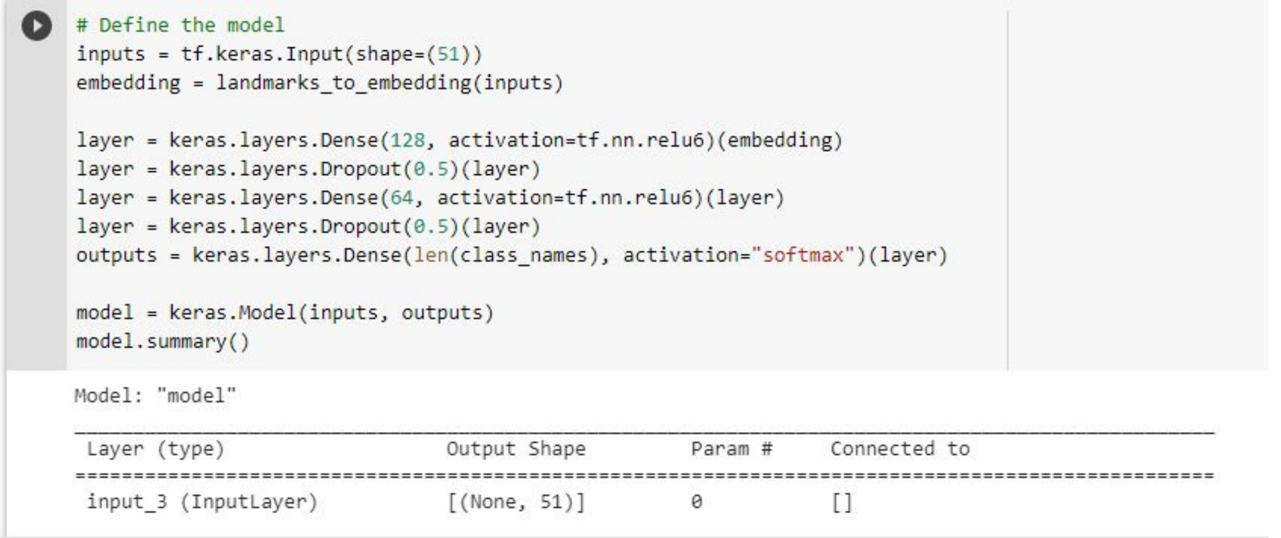
Epochs: 30/200

Optimization:

Dropout Layers

Early Stopping

Loss: 0.19, Accuracy: 0.94

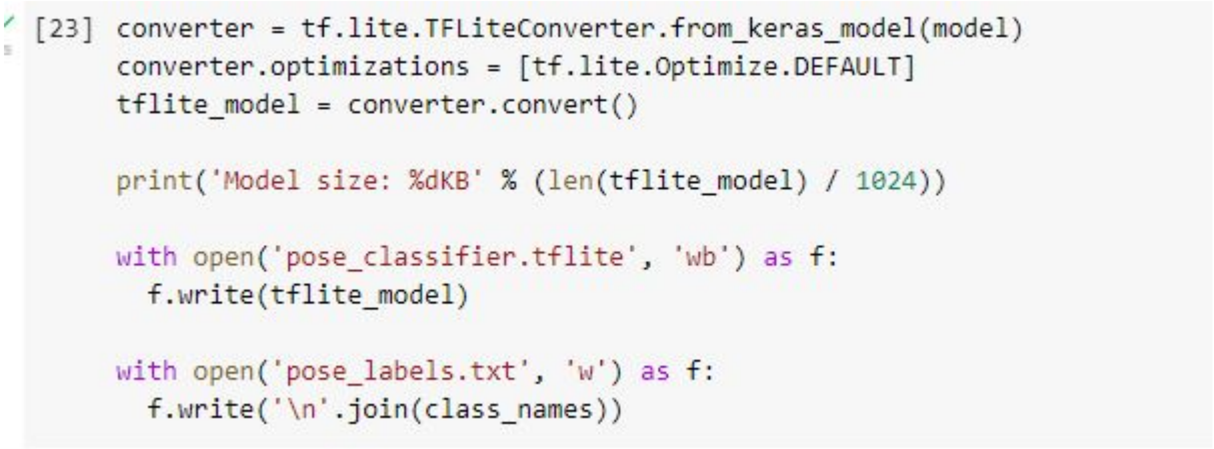


```
# Define the model
inputs = tf.keras.Input(shape=(51))
embedding = landmarks_to_embedding(inputs)

layer = keras.layers.Dense(128, activation=tf.nn.relu6)(embedding)
layer = keras.layers.Dropout(0.5)(layer)
layer = keras.layers.Dense(64, activation=tf.nn.relu6)(layer)
layer = keras.layers.Dropout(0.5)(layer)
outputs = keras.layers.Dense(len(class_names), activation="softmax")(layer)

model = keras.Model(inputs, outputs)
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_3 (InputLayer)           [(None, 51)]         0           []
```

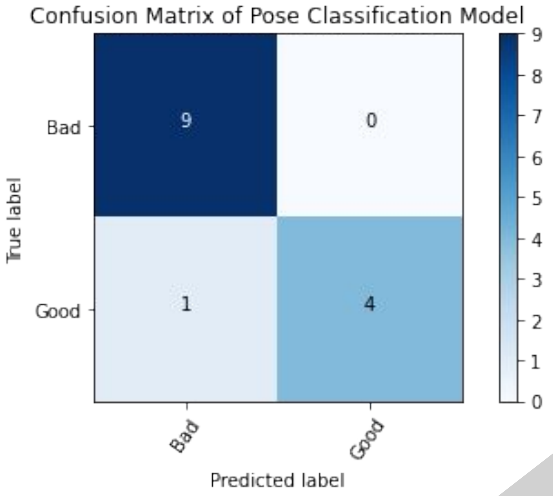
MoveNet

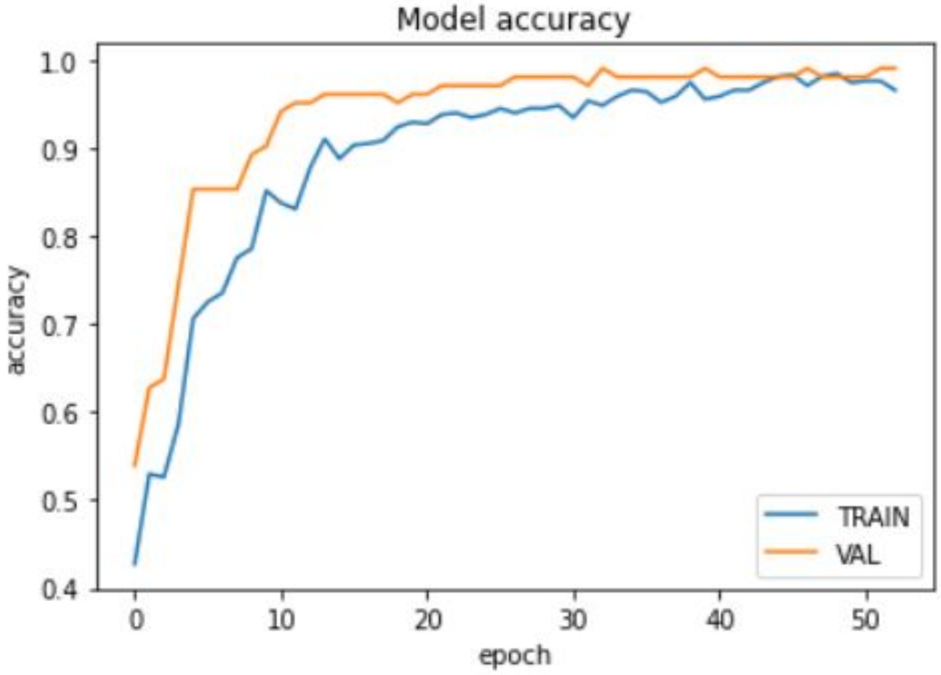
Once trained, we convert model to tflite



```
[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))
```

Results





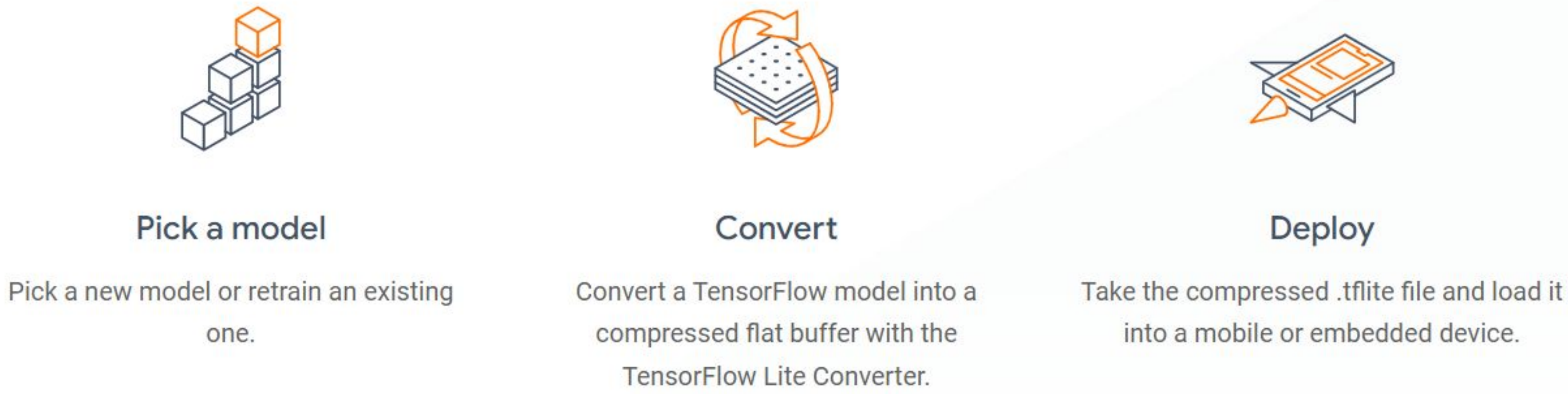
```
Classification Report:
              precision    recall  f1-score   support

         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
```

WEB-APP

Problem Statement: To run a Application on client end

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.



Pick a model

Pick a new model or retrain an existing one.

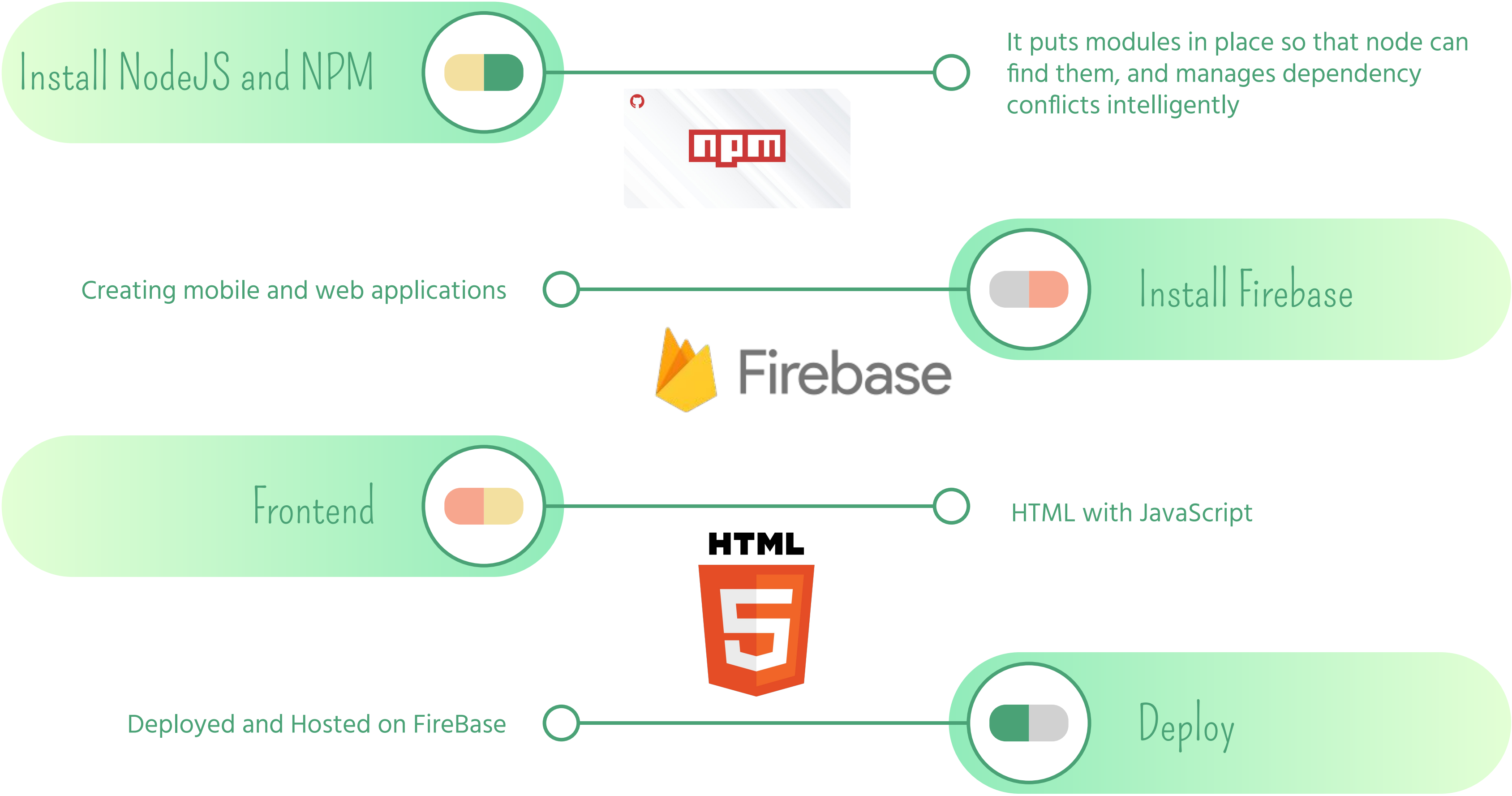
Convert

Convert a TensorFlow model into a compressed flat buffer with the TensorFlow Lite Converter.

Deploy

Take the compressed .tflite file and load it into a mobile or embedded device.

DEPLOYMENT



Install NodeJS and NPM

It puts modules in place so that node can find them, and manages dependency conflicts intelligently

Creating mobile and web applications

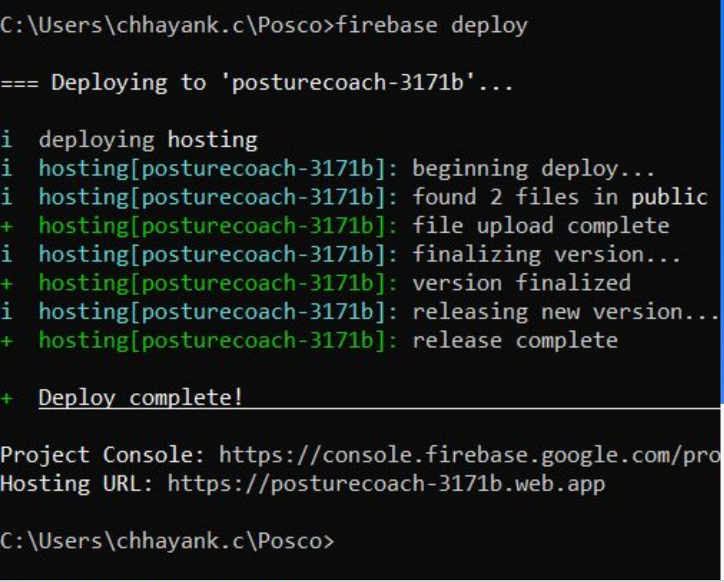
Install Firebase

Frontend

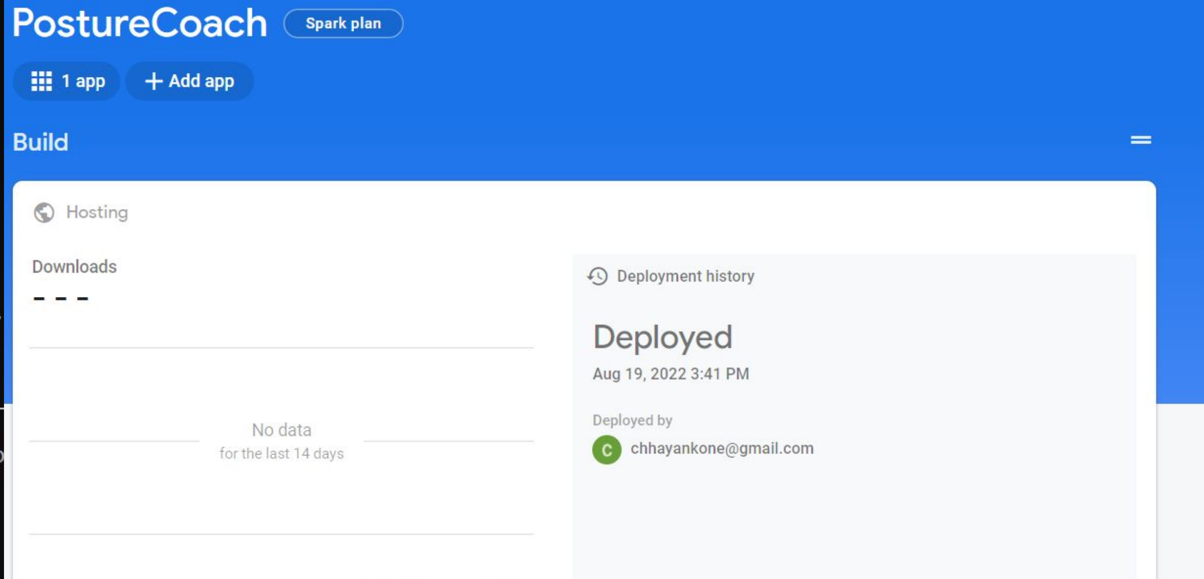
HTML with JavaScript

Deployed and Hosted on FireBase

Deploy



```
C:\Users\chhayank.c\Posco>firebase deploy

=== Deploying to 'posturecoach-3171b'...

i  deploying hosting
i  hosting[posturecoach-3171b]: beginning deploy...
i  hosting[posturecoach-3171b]: found 2 files in public
+  hosting[posturecoach-3171b]: file upload complete
i  hosting[posturecoach-3171b]: finalizing version...
+  hosting[posturecoach-3171b]: version finalized
i  hosting[posturecoach-3171b]: releasing new version...
+  hosting[posturecoach-3171b]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/pro
Hosting URL: https://posturecoach-3171b.web.app

C:\Users\chhayank.c\Posco>
```



PostureCoach Spark plan

1 app + Add app

Build

Hosting

Downloads

- - -

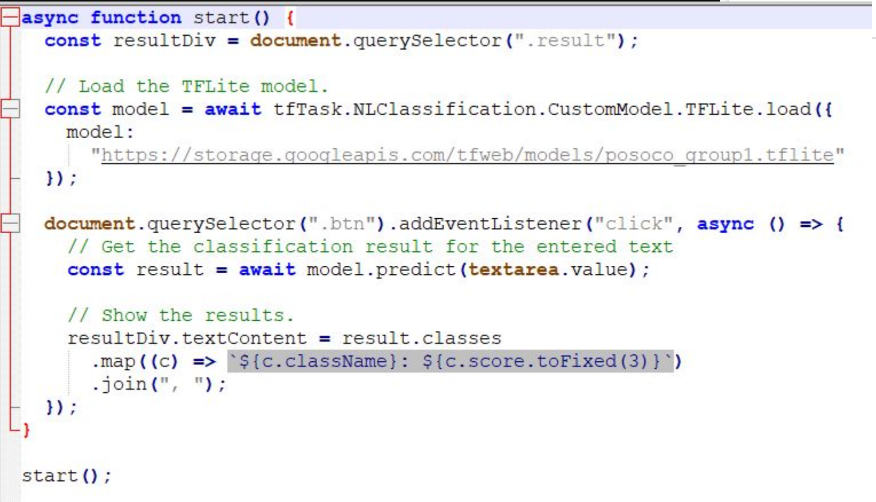
No data
for the last 14 days

— This week - - Last week

Deployment history

Deployed
Aug 19, 2022 3:41 PM

Deployed by
C chhayankone@gmail.com



```
async function start() {
  const resultDiv = document.querySelector(".result");

  // Load the TFLite model.
  const model = await tfTask.NLClassification.CustomModel.TFLite.load({
    model:
      "https://storage.googleapis.com/tfweb/models/posoco_group1.tflite"
  });

  document.querySelector(".btn").addEventListener("click", async () => {
    // Get the classification result for the entered text
    const result = await model.predict(textarea.value);

    // Show the results.
    resultDiv.textContent = result.classes
      .map((c) => `${c.className}: ${c.score.toFixed(3)}`)
      .join(", ");
  });
}

start();
```

Results

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

Overall Accuracy ranges between 92-95%
Overall Loss ranges between 0.19 - 0.12

Future Scope

- Privacy concerns
- The dataset needs to be more inclusive of all physiques
- Audio alerts when bad posture detected
- Inclusion of medically validated training dataset

DEMO

POSCO

POSTURE COACH

Thanks

Do you have any questions?