Python les-materialen

# Overview

This Colab helps to create a training set for the k-NN classifier described in the MediaPipe [Pose Classification](https://google.github.io/mediapipe/solutions/pose_classification.html) soultion, export it to a CSV and then use it in the [ML Kit sample app](https://developers.google.com/ml-kit/vision/pose-detection/classifying-poses#4_integrate_with_the_ml_kit_quickstart_app).

# Step 0: Start Colab

Connect the Colab to hosted Python3 runtime (check top-right corner) and then install required dependencies.

!pip install numpy==1.19.3  
!pip install opencv-python==4.5.1.48  
!pip install tqdm==4.56.0  
  
!pip install mediapipe==0.8.3

# Step 1: Upload image samples

Locally create a folder named fitness\_poses\_images\_in with image samples.

Images should repesent terminal states of desired pose classes. I.e. if you want to classify push-up provide iamges for two classes: when person is up, and when person is down.

There should be about a few hundred samples per class covering different camera angles, environment conditions, body shapes, and exercise variations to build a good classifier.

Required structure of the images\_in\_folder:

fitness\_poses\_images\_in/  
 pushups\_up/  
 image\_001.jpg  
 image\_002.jpg  
 ...  
 pushups\_down/  
 image\_001.jpg  
 image\_002.jpg  
 ...  
 ...

Zip the fitness\_poses\_images\_in folder:

zip -r fitness\_poses\_images\_in.zip fitness\_poses\_images\_in

And run the code below to upload it to the Colab runtime

from google.colab import files  
import os  
  
uploaded = files.upload()  
os.listdir('.')

Unzip the archive:

import zipfile  
import io  
  
zf = zipfile.ZipFile(io.BytesIO(uploaded['fitness\_poses\_images\_in.zip']), "r")  
zf.extractall()  
os.listdir('.')

# Step 2: Create samples for classifier

Runs BlazePose on provided images to get target poses for the classifier in a format required by the demo App.

# Folder with images to use as target poses for classification.  
#  
# Images should repesent terminal states of desired pose classes. I.e. if you  
# want to classify push-up provide iamges for two classes: when person is up,  
# and when person is down.  
#  
# Required structure of the images\_in\_folder:  
# fitness\_poses\_images\_in/  
# pushups\_up/  
# image\_001.jpg  
# image\_002.jpg  
# ...  
# pushups\_down/  
# image\_001.jpg  
# image\_002.jpg  
# ...  
# ...  
images\_in\_folder = 'fitness\_poses\_images\_in'  
  
# Output folders for bootstrapped images and CSVs. Image will have a predicted  
# Pose rendering and can be used to remove unwanted samples.  
images\_out\_folder = 'fitness\_poses\_images\_out\_basic'  
  
# Output CSV path to put bootstrapped poses to. This CSV will be used by the  
# demo App.  
#  
# Output CSV format:  
# sample\_00001,pose\_class\_1,x1,y1,z1,x2,y2,z2,...,x33,y33,z33  
# sample\_00002,pose\_class\_2,x1,y1,z1,x2,y2,z2,...,x33,y33,z33  
# ...  
#  
csv\_out\_path = 'fitness\_poses\_csvs\_out\_basic.csv'

import csv  
import cv2  
import numpy as np  
import os  
import sys  
import tqdm  
  
from mediapipe.python.solutions import drawing\_utils as mp\_drawing  
from mediapipe.python.solutions import pose as mp\_pose  
  
  
with open(csv\_out\_path, 'w') as csv\_out\_file:  
 csv\_out\_writer = csv.writer(csv\_out\_file, delimiter=',', quoting=csv.QUOTE\_MINIMAL)  
  
 # Folder names are used as pose class names.  
 pose\_class\_names = sorted([n for n in os.listdir(images\_in\_folder) if not n.startswith('.')])  
  
 for pose\_class\_name in pose\_class\_names:  
 print('Bootstrapping ', pose\_class\_name, file=sys.stderr)  
  
 if not os.path.exists(os.path.join(images\_out\_folder, pose\_class\_name)):  
 os.makedirs(os.path.join(images\_out\_folder, pose\_class\_name))  
  
 image\_names = sorted([  
 n for n in os.listdir(os.path.join(images\_in\_folder, pose\_class\_name))  
 if not n.startswith('.')])  
 for image\_name in tqdm.tqdm(image\_names, position=0):  
 # Load image.  
 input\_frame = cv2.imread(os.path.join(images\_in\_folder, pose\_class\_name, image\_name))  
 input\_frame = cv2.cvtColor(input\_frame, cv2.COLOR\_BGR2RGB)  
  
 # Initialize fresh pose tracker and run it.  
 with mp\_pose.Pose(upper\_body\_only=False) as pose\_tracker:  
 result = pose\_tracker.process(image=input\_frame)  
 pose\_landmarks = result.pose\_landmarks  
   
 # Save image with pose prediction (if pose was detected).  
 output\_frame = input\_frame.copy()  
 if pose\_landmarks is not None:  
 mp\_drawing.draw\_landmarks(  
 image=output\_frame,  
 landmark\_list=pose\_landmarks,  
 connections=mp\_pose.POSE\_CONNECTIONS)  
 output\_frame = cv2.cvtColor(output\_frame, cv2.COLOR\_RGB2BGR)  
 cv2.imwrite(os.path.join(images\_out\_folder, image\_name), output\_frame)  
   
 # Save landmarks.  
 if pose\_landmarks is not None:  
 # Check the number of landmarks and take pose landmarks.  
 assert len(pose\_landmarks.landmark) == 33, 'Unexpected number of predicted pose landmarks: {}'.format(len(pose\_landmarks.landmark))  
 pose\_landmarks = [[lmk.x, lmk.y, lmk.z] for lmk in pose\_landmarks.landmark]  
  
 # Map pose landmarks from [0, 1] range to absolute coordinates to get  
 # correct aspect ratio.  
 frame\_height, frame\_width = output\_frame.shape[:2]  
 pose\_landmarks \*= np.array([frame\_width, frame\_height, frame\_width])  
  
 # Write pose sample to CSV.  
 pose\_landmarks = np.around(pose\_landmarks, 5).flatten().astype(np.str).tolist()  
 csv\_out\_writer.writerow([image\_name, pose\_class\_name] + pose\_landmarks)

Now look at the output images with predicted Pose and remove those you are not satisfied with from the output CSV. Wrongly predicted poses will affect accuracy of the classification.

Once done, you can use the CSV in the demo App.

For more accurate validation of the predicted Poses use extended Colab provided in the documentation.

# Step 3: Download CSV

Please check this [guide](https://developers.google.com/ml-kit/vision/pose-detection/classifying-poses#4_integrate_with_the_ml_kit_quickstart_app) on how to use this CSV in the ML Kit sample app.

files.download(csv\_out\_path)