Overview

This Colab helps to create a training set for the k-NN classifier described in the MediaPipe Pose Classification soultion, export it to a CSV and then use it in the ML Kit sample 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 on how to use this CSV in the ML Kit sample app.

```
files.download(csv_out_path)
```