Jupyter Notebook code used to generate dataset csv files from raw images from the asl dataset

In [ ]:
```
import cv2
import time
import mediapipe
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from PIL import Image, ImageOps
import pandas as pd
import matplotlib.pyplot as plt
import os
from os import listdir
import numpy as np
```

In [ ]:
```
# generate a feature vector (1x40) from the array of coordinates of the 21 hand landmarks

def generateFeatures(coords):
    relative = []
    x0, y0 = coords[0]
    x1, y1 = coords[1]
    d = np.sqrt((x1-x0)*(x1-x0)+(y1-y0)*(y1-y0))
    for i in range(20):
        x1, y1 = coords[i+1]
        relative.append((x1-x0)/d) #relative and normalized distance to landmark 0 (base of palm)
        relative.append((y1-y0)/d)
    return np.array(relative)
```

In [ ]:
```
# Generate a dataset of feature vectors from the downloaded images f

drawingModule = mediapipe.solutions.drawing_utils
handsModule = mediapipe.solutions.hands
hands =  handsModule.Hands(static_image_mode=True, min_detection_confidence=0.7, min_tracking_confidence=0.7, max_num_hands=1)

train1 = []
test1 = []
for i in range(10):
    dir = "dataset/asl_dataset/" + str(i) #process all the number images (0 - 9 digits)
    counter = 0 # cycles through 0 to 9 for each label
    for item in os.listdir(dir): # loop through each image in each directory
        img = Image.open(dir + "/" + item).resize((100,100))
        img = np.array(img.getdata(), dtype='uint8')
        img = np.resize(img, (100, 100, 3))
        results = hands.process(img) # use mediapipe to extract hand landmarks from raw dataset image
        if results.multi_hand_landmarks != None:
            for handLandmarks in results.multi_hand_landmarks:
                drawingModule.draw_landmarks(img, handLandmarks, handsModule.HAND_CONNECTIONS)
                coords = []
                found = 0
                for point in handsModule.HandLandmark:
                    normalizedLandmark = handLandmarks.landmark[point]
                    pixelCoordinatesLandmark= drawingModule._normalized_to_pixel_coordinates(normalizedLandmark.x, normalizedLandmark.y, 640, 480)
                    if pixelCoordinatesLandmark:
                        coords.append(pixelCoordinatesLandmark)
                        found += 1
                if found==21: # only use images where mediapipe can find all 21 landmarks
                    arr = generateFeatures(coords)
                    arr = np.insert(arr, 0, i) # add the label to the front of the feature vector
                    # split 80% to training set and 20% to validation set
                    if(counter<8):
                        train1.append(arr) # add the entry to an array to be saved to csv later
                    else: test1.append(arr)
                    if(counter==9): counter = 0
                    else: counter = counter + 1
```

In [ ]:
```
# save the generated numbers dataset to csv files for easy importing later

dftrain1 = pd.DataFrame(train1)
dftest1 = pd.DataFrame(test1)
dftrain1.to_csv("dataset/asl_dataset/numbers_mp_train.csv", index=False)
dftest1.to_csv("dataset/asl_dataset/numbers_mp_test.csv", index=False)
```

In [ ]:
```
# do the same for all letters (except j and z)

train2 = []
test2 = []
for i in range(25): #25 instead of 26 to omit z
    if(i==9): continue # omit j
    dir = "dataset/asl_dataset/" + chr(ord('a')+i)
    counter = 0
    for item in os.listdir(dir):
        img = Image.open(dir + "/" + item).resize((100,100))
        img = np.array(img.getdata(), dtype='uint8')
        img = np.resize(img, (100, 100, 3))
        results = hands.process(img)
        if results.multi_hand_landmarks != None:
            for handLandmarks in results.multi_hand_landmarks:
                drawingModule.draw_landmarks(img, handLandmarks, handsModule.HAND_CONNECTIONS)
                coords = []
                found = 0
                for point in handsModule.HandLandmark:
                    normalizedLandmark = handLandmarks.landmark[point]
                    pixelCoordinatesLandmark= drawingModule._normalized_to_pixel_coordinates(normalizedLandmark.x, normalizedLandmark.y, 640, 480)
                    if pixelCoordinatesLandmark:
                        coords.append(pixelCoordinatesLandmark)
                        found += 1
                if found==21:
                    arr = generateFeatures(coords)
                    arr = np.insert(arr, 0, i)
                    counter = counter + 1
                    if(counter<8):
                        train2.append(arr)
                    else: test2.append(arr)
                    if(counter==9): counter = 0
                    else: counter = counter + 1
```

In [ ]:
```
# save the generated letters dataset to csv files for easy importing later

dftrain2 = pd.DataFrame(train2)
dftest2 = pd.DataFrame(test2)
dftrain2.to_csv("dataset/asl_dataset/letters_mp_train.csv", index=False)
dftest2.to_csv("dataset/asl_dataset/letters_mp_test.csv", index=False)
```