به نام خدا

پروژه دوم

محمد محسن همایونی

در ابتدا کتاب خانه های مورد نیاز را نصب میکنیم

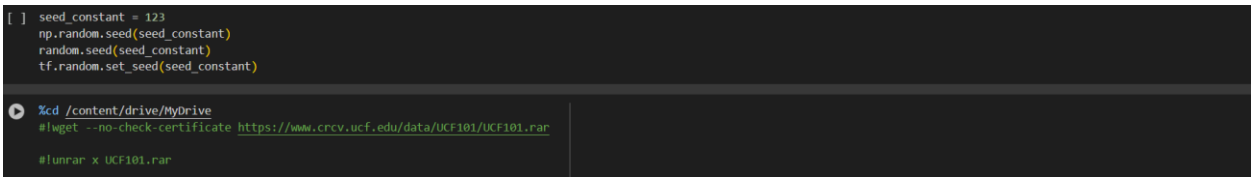
```
from google.colab import drive
drive.mount('/content/drive')
!pip install tensorflow

import os
import cv2
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
from collections import deque
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
```

سپس برای ثابت ماندن وزن ها seed استفاده میکنیم ودیتا ست را دانلود میکنیم:



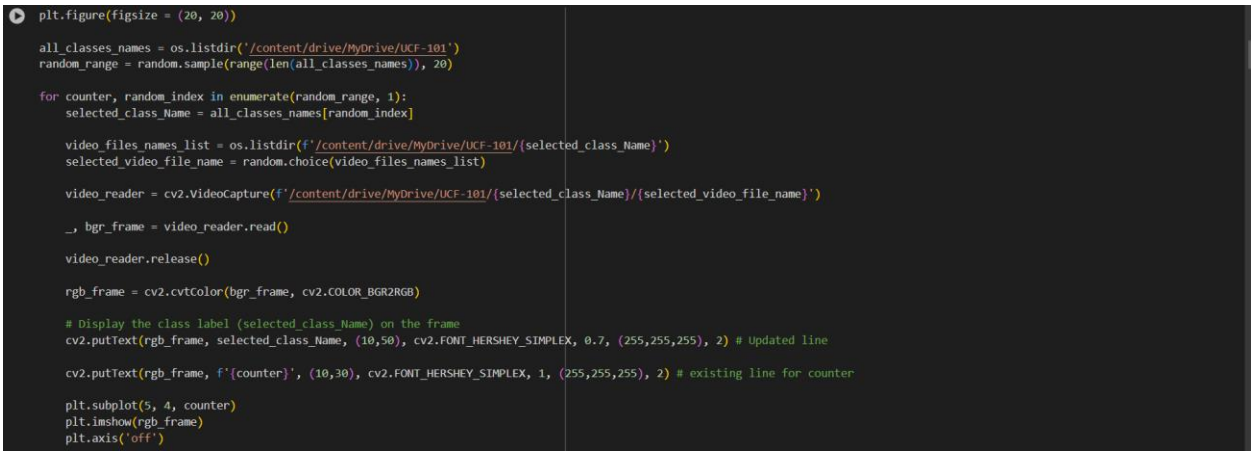
```
seed_constant = 123
np.random.seed(seed_constant)
random.seed(seed_constant)
tf.random.set_seed(seed_constant)
```



```
%cd /content/drive/MyDrive
#!wget --no-check-certificate https://www.crcv.ucf.edu/data/UCF101/UCF101.rar

#!unrar x UCF101.rar
```

در بخش بعدی هدف ما انتخاب ۲۰ فریم تصادفی از کلاس های مختلف است تا بررسی کنیم که آیا درست کلاس بندی شده اند یانه؟



```
plt.figure(figsize = (20, 20))

all_classes_names = os.listdir('/content/drive/MyDrive/UCF-101')
random_range = random.sample(range(len(all_classes_names)), 20)

for counter, random_index in enumerate(random_range, 1):
    selected_class_Name = all_classes_names[random_index]

    video_files_names_list = os.listdir(f'/content/drive/MyDrive/UCF-101/{selected_class_Name}')
    selected_video_file_name = random.choice(video_files_names_list)

    video_reader = cv2.VideoCapture(f'/content/drive/MyDrive/UCF-101/{selected_class_Name}/{selected_video_file_name}')

    _, bgr_frame = video_reader.read()

    video_reader.release()

    rgb_frame = cv2.cvtColor(bgr_frame, cv2.COLOR_BGR2RGB)

    # Display the class label (selected_class_Name) on the frame
    cv2.putText(rgb_frame, selected_class_Name, (10,50), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,255), 2) # Updated line

    cv2.putText(rgb_frame, f'{counter}', (10,30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2) # existing line for counter

    plt.subplot(5, 4, counter)
    plt.imshow(rgb_frame)
    plt.axis('off')
```

سپس ابعاد ورودی وتعدادی کلاس را انتخاب میکنیم تا آماده پردازش و دادن به مدل شوند. در این مساله برای راحتی ۳ کلاس انتخاب کردم.

در مرحله بعدی تابعی میسازیم تا فریم هایی را استخراج کند:

```python
def frames_extraction(video_path):
    frames_list = []

    video_reader = cv2.VideoCapture(video_path)

    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    skip_frames_window = max(int(video_frames_count/sequence_length), 1)

    for frame_counter in range(sequence_length):

        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)

        success, frame = video_reader.read()

        if not success:
            break

        resized_frame = cv2.resize(frame, (image_height, image_width))

        normalized_frame = resized_frame / 255

        frames_list.append(normalized_frame)

    video_reader.release()
```

در تابع بعدی ما دیتا ستی برای آموزش میسازیم که ویدیو ها را استخراج میکند کلاس را به صورت ایندکس برچسب گذاری میکند و ویژگی ها را استخراج میکند:

```python
def create_dataset():
    features = []
    labels = []
    video_files_paths = []

    for class_index, class_name in enumerate(Classes_list):
        print(f'Extracting Data of Class: {class_name}')

        files_list = os.listdir(os.path.join(DATASET_DIR, class_name))

        for file_name in files_list:
            video_file_path = os.path.join(DATASET_DIR, class_name, file_name)

            frames = frames_extraction(video_file_path)

            if len(frames) == sequence_length:
                features.append(frames)
                labels.append(class_index)
                video_files_paths.append(video_file_path)

    # Return statement moved outside the loop
    features = np.asarray(features)
```

به صورت زیر مدل خود را طراحی میکنیم:

```python
def create_covLstm_model():
    model = Sequential()

    model.add(ConvLSTM2D(filters=4, kernel_size=(3, 3), activation='tanh',
                         recurrent_dropout=0.2, return_sequences=True,
                         input_shape=(sequence_length, image_height, image_width, 3)))
    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='channels_last'))
    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters=8, kernel_size=(3, 3), activation='tanh',
                         recurrent_dropout=0.2, return_sequences=True))
    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='channels_last'))
    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters=14, kernel_size=(3, 3), activation='tanh',
                         recurrent_dropout=0.2, return_sequences=True))
    model.add(MaxPooling3D(pool_size=(1, 2, 2), padding='same', data_format='channels_last'))
    model.add(TimeDistributed(Dropout(0.2)))

    model.add(ConvLSTM2D(filters=16, kernel_size=(3, 3), activation='tanh',
                         recurrent_dropout=0.2, return_sequences=False))  # return_sequences=False here

    # Remove the 3D pooling since the tensor is now 4D
    model.add(Flatten())  # Flatten the 4D output
    model.add(Dense(len(Classes_list), activation='softmax'))
```

در انتها شبکه را ترین کرده و نمودار های آن را رسم میکنیم.

برای تست ویدیو من تابعی طراحی کردم که با دانلود کردن ویدیو از یوتیوب و اعمال کار های دیتایی روی آن (جداکردن فریم ها و ویژگی ها) میتواند کلاس آن را تشخیص دهد.