SUMMER INTERNSHIP PROJECT ON VIDEO CLASSIFICATION

BY

K. KEERTHI HUMSIKA(N161299)

M.SRAVANI(N160564)

V.JOHNY(N160476)

A. JAGADEESH KUMAR(N160179)

CH. AJAY KUMAR(N160219)

L.VIKASH(N160098)

Under the guidance of

Nagarjuna Devi Madam

Faculty, Department of CSE

APIIIT, RGUKT Nuzvid

A Project Work submitted to the Department of CSE for Summer Internship project of E3-SEM-2.

Department of Computer Science and Engineering

CERTIFICATE OF PROJECT COMPLETION This is to certify that the project work entitled “Video Classification” is a bonafide work by N161299, N160564, N160476, N160179, N160219 and N160098 submitted for E3 SEM1 mini project to the department of CSE under my guidance during the academic year 2020-2021. This project in my opinion, is worthy of consideration for the awarding of marks in accordance with the Department and University regulations.

Date:

Place:

Kumar Anurupam Sir	**Supervisor**
Head of the Department	**Nagarjuna Devi Madam**
Department of Computer Science	**Faculty**
and Engineering	**Department of CSE**

ABSTRACT:

Today's rapidly growing internet era has also increased the data in a very large amount for which the classification of the data has become more Important than ever before. As daily millions of video are been uploaded on the internet usually it is being classified based on the title of the video, which is not enough as the content of the video may be different than that of the title. Thus, this project highlights a system that would classify the video based on real-time data which takes the content of the video as the input and accordingly it gives the output. The major crux of this system is the usage of deep learning, in which the collection of the image is been taken from the video and compared with the pre-trained model and accordingly the result showed.

EXISTING AND PROPOSAL SYSTEMS:

➢ Existing System:

In existing system, we have a problem called 'Flickering' where the activity name in the output gets fluctuated.

➢ Proposed System:

In present system, we have eradicated the problem of 'Flickering' by using efficient method called Convolution Neural Networks (CNN).

TECHNOLOGIES

- Python is the programming language that we use to develop this project.
- The main domain that we chose to develop this project is Deep-learning.
- The reason we choose the python language is it is a platform independent and it has many inbuilt packages to implement this project.

LIBRARIES USED IN THE PROJECT:

- Numpy
- Keras
- Open CV2
- Tensor flow
- Sklearn
- Pickle

Numpy:

- Numpy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.
- Numpy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

Keras:

- Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models.
- It wraps the efficient numerical computation libraries Theano and Tensor Flow and allows you to define and train neural network models in just a few lines of code.

Open CV2:

- OpenCV is the huge open-source library for the computer vision**,** machine learning**,** and image processing and now it plays a major role in real-time operation which is very important in today's systems.

- By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

Tensor Flow:

- TensorFlow is a Python library for fast numerical computing created and released by Google.
- It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of Tensor Flow.

Sklearn:

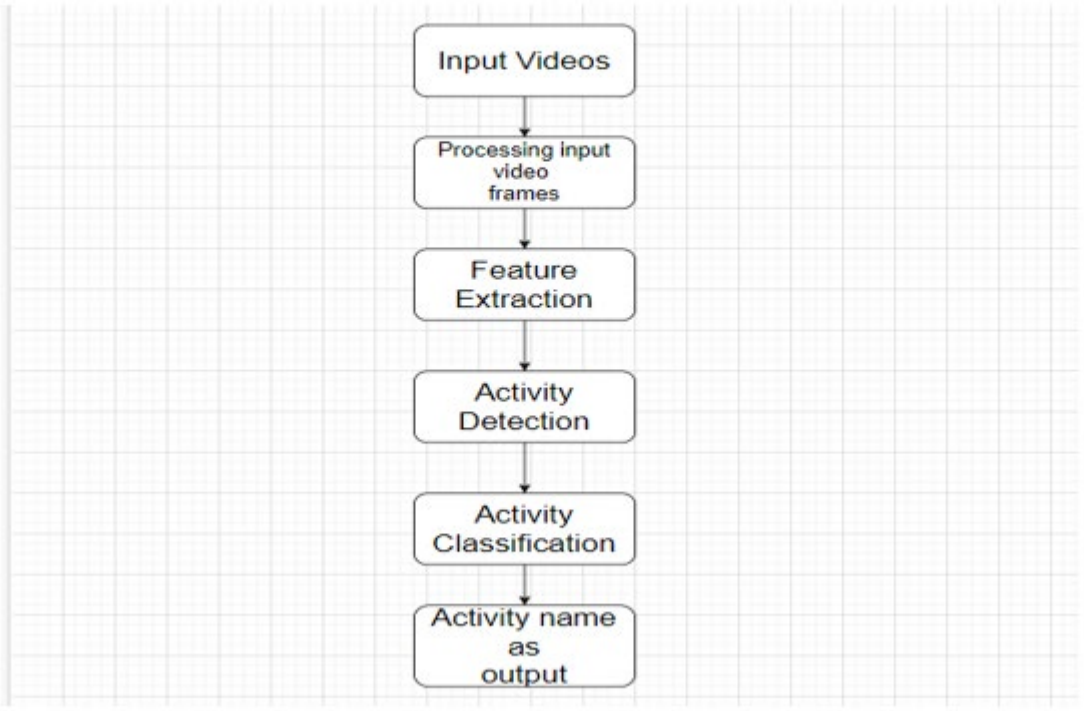
- Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like Numpy, pandas, and Matplotlib.

Pickle:

- Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's

the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.

Flow chart for Video Classification:



➢ Requirements:

Software requirements:

1. Visual studio
2. Windows 10.

Hardware requirements:

1. PC with minimum 16GB RAM
2. i7 with 64-bit processor.

When performing image classification, we:

1. Input an image to our CNN
2. Obtain the predictions from the CNN
3. Choose the label with the largest corresponding probability

Since a video is just a series of frames, a naive video classification method would be to:

1. Loop over all frames in the video file
2. For each frame, pass the frame through the CNN
3. Classify each frame *individually* and *independently* of each other
4. Choose the label with the largest corresponding probability
5. Label the frame and write the output frame to disk

There's a problem with this approach though — if you've ever tried to apply simple image classification to video classification you likely encountered a sort of **"prediction flickering"** as seen in the video at the top of this section. Notice how in this visualization we see our CNN shifting between two predictions: *"football"* and the correct label, *"weightlifting"*.

The video is clearly of weightlifting and we would like our entire video to be labelled as such — but how we can prevent the CNN "flickering" between these two labels?

A simple, yet elegant solution, is to utilize a rolling prediction average.

Our algorithm now becomes:

1. Loop over all frames in the video file
2. For each frame, pass the frame through the CNN
3. Obtain the predictions from the CNN
4. Maintain a list of the last *K* predictions
5. Compute the average of the last *K* predictions and choose the label with the largest corresponding probability
6. Label the frame and write the output frame to disk

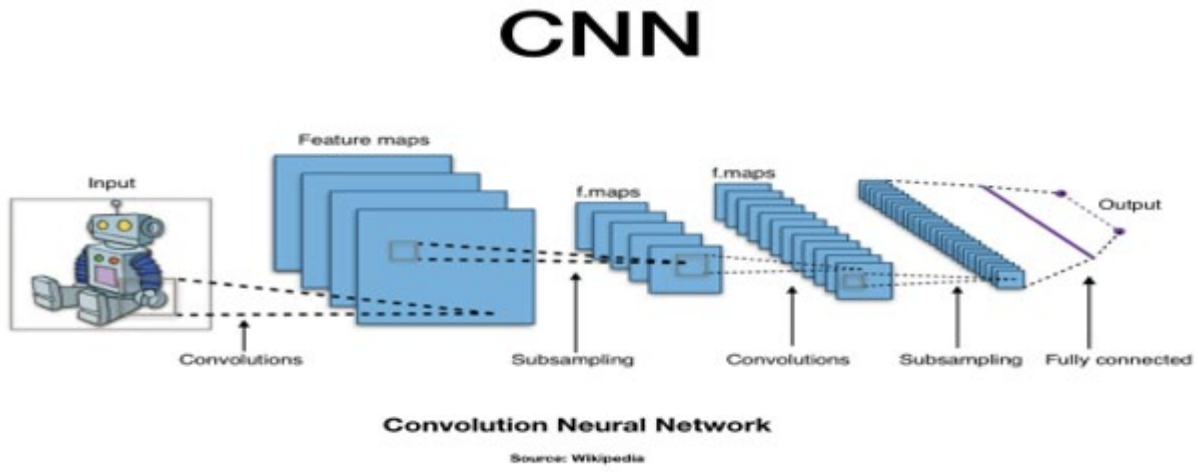
<u>To classify the Frames, we use Convolutional neural networks (CNN):</u>

When performing image classification, we do:

- ▶ Input an image to our CNN.
- ▶ Obtain the predictions from the CNN.
- ▶ Choose the label with the largest corresponding probability.

- A video is nothing but series of frames, where CNN model helps in dividing the video into N – number of frames.

- Now the model tries to match that particular frame with the trained data sets and gives the corresponding name as output.



The convolutional neural network that is CNN is a machine learning algorithm and it having different convolutional layers that are input and output layers and many hidden layers that are used for image classification, image processing, other different auto correlated data and also for segmentation. The CNN convolutional neural network is commonly used for image or video classification.

Implementing Keras training script:

Let's go ahead and implement our training script used to train a Keras CNN to recognize each of the sports activities.

Open up the train.py file and insert the following code:

```
1.  # set the matplotlib backend so figures can be saved in the background
2.  import matplotlib
3.  matplotlib.use("Agg")
4.
5.  # import the necessary packages
6.  from tensorflow.keras.preprocessing.image import ImageDataGenerator
7.  from tensorflow.keras.layers import AveragePooling2D
8.  from tensorflow.keras.applications import ResNet50
9.  from tensorflow.keras.layers import Dropout
10. from tensorflow.keras.layers import Flatten
11. from tensorflow.keras.layers import Dense
12. from tensorflow.keras.layers import Input
13. from tensorflow.keras.models import Model
14. from tensorflow.keras.optimizers import SGD
15. from sklearn.preprocessing import LabelBinarizer
16. from sklearn.model_selection import train_test_split
17. from sklearn.metrics import classification_report
18. from imutils import paths
19. import matplotlib.pyplot as plt
20. import numpy as np
21. import argparse
22. import pickle
23. import cv2
24. import os
```

On **Lines 2-24**, we import necessary packages for training our classifier:

```
26. # construct the argument parser and parse the arguments
27. ap = argparse.ArgumentParser()
28. ap.add_argument("-d", "--dataset", required=True,
29.     help="path to input dataset")
30. ap.add_argument("-m", "--model", required=True,
31.     help="path to output serialized model")
32. ap.add_argument("-l", "--label-bin", required=True,
33.     help="path to output label binarizer")
34. ap.add_argument("-e", "--epochs", type=int, default=25,
35.     help="# of epochs to train our network for")
36. ap.add_argument("-p", "--plot", type=str, default="plot.png",
37.     help="path to output loss/accuracy plot")
38. args = vars(ap.parse_args())
```

Our script accepts five command line arguments, the first three of which are required:

- --dataset

  : The path to the input dataset.

- --model

  : Our path to our output Keras model file.

- --label-bin

```
40. | # initialize the set of labels from the spots activity dataset we are
41. | # going to train our network on
42. | LABELS = set(["weight_lifting", "tennis", "football"])
43. |
44. | # grab the list of images in our dataset directory, then initialize
45. | # the list of data (i.e., images) and class images
46. | print("[INFO] loading images...")
47. | imagePaths = list(paths.list_images(args["dataset"]))
48. | data = []
49. | labels = []
50. |
51. | # loop over the image paths
52. | for imagePath in imagePaths:
53. |     # extract the class label from the filename
54. |     label = imagePath.split(os.path.sep)[-2]
55. |
56. |     # if the label of the current image is not part of of the labels
57. |     # are interested in, then ignore the image
58. |     if label not in LABELS:
59. |         continue
60. |
61. |     # load the image, convert it to RGB channel ordering, and resize
62. |     # it to be a fixed 224x224 pixels, ignoring aspect ratio
63. |     image = cv2.imread(imagePath)
64. |     image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
65. |     image = cv2.resize(image, (224, 224))
66. |
67. |     # update the data and labels lists, respectively
68. |     data.append(image)
69. |     labels.append(label)
```

Line 42 contains the set of class labels for which our dataset will consist of. All labels *not* present in this set will be *excluded* from being part of our dataset. To save on training time, our dataset will only consist of *weight lifting, tennis,* and *football/soccer*. Feel free to work with other classes by making changes to the lables set.

```
61.      # make predictions on the frame and then update the predictions
62.      # queue
63.      preds = model.predict(np.expand_dims(frame, axis=0))[0]
64.      Q.append(preds)
65.
66.      # perform prediction averaging over the current history of
67.      # previous predictions
68.      results = np.array(Q).mean(axis=0)
69.      i = np.argmax(results)
70.      label = lb.classes_[i]
```

Line 63 makes predictions on the *current* frame. The prediction results are added to the via Line 64.

```
72.      # draw the activity on the output frame
73.      text = "activity: {}".format(label)
74.      cv2.putText(output, text, (35, 50), cv2.FONT_HERSHEY_SIMPLEX,
75.          1.25, (0, 255, 0), 5)
```

Lines 73-75 draw the prediction on the frame gives output.

```
78.      if writer is None:
79.          # initialize our video writer
80.          fourcc = cv2.VideoWriter_fourcc(*"MJPG")
81.          writer = cv2.VideoWriter(args["output"], fourcc, 30,
82.              (W, H), True)
```

Initialize the video if necessary. The frame of the output is written to the file.

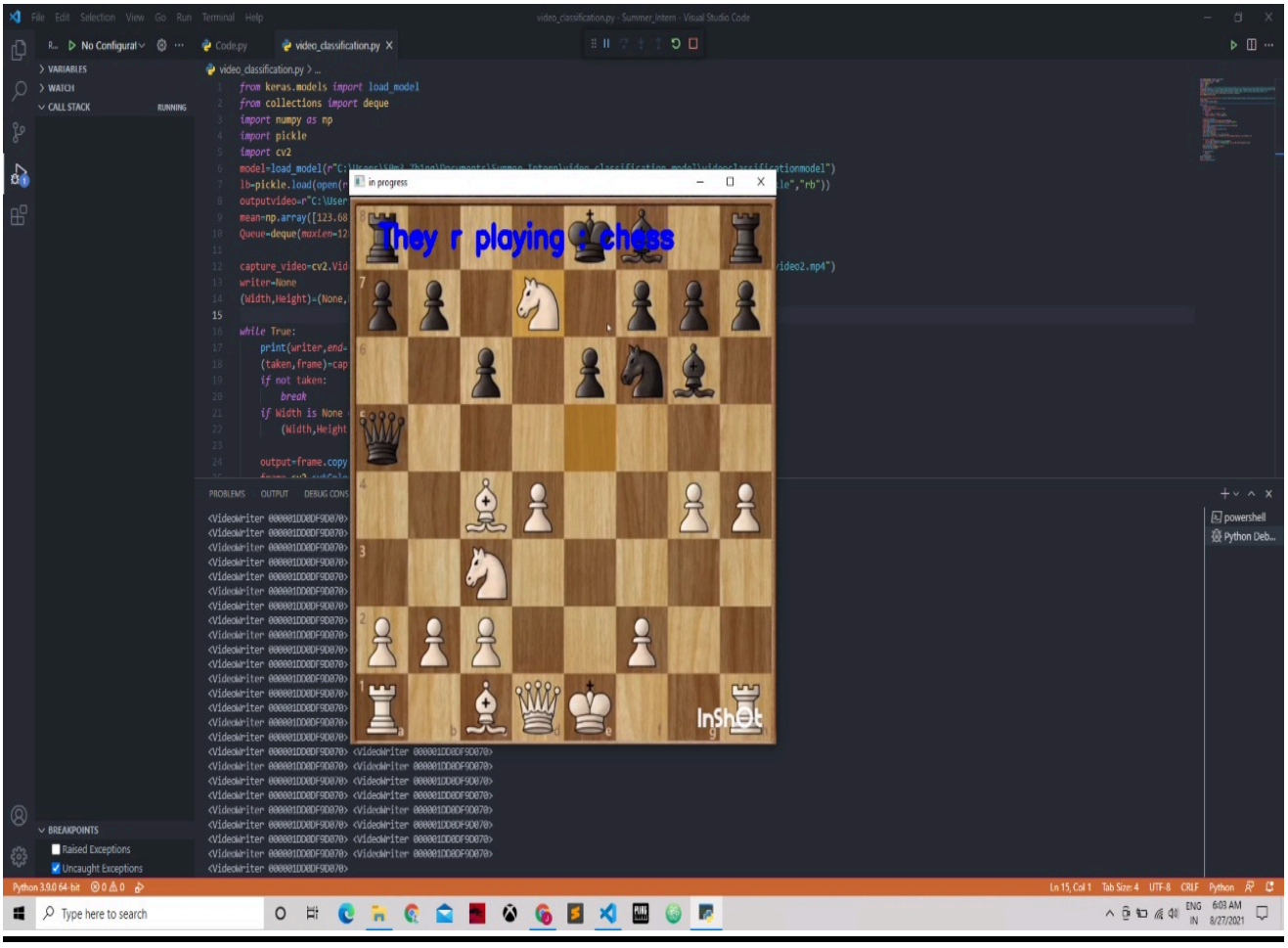
We give the above image as input to CNN model. Then it displays the output as corresponding activity name.

The model displays the activity name as football.

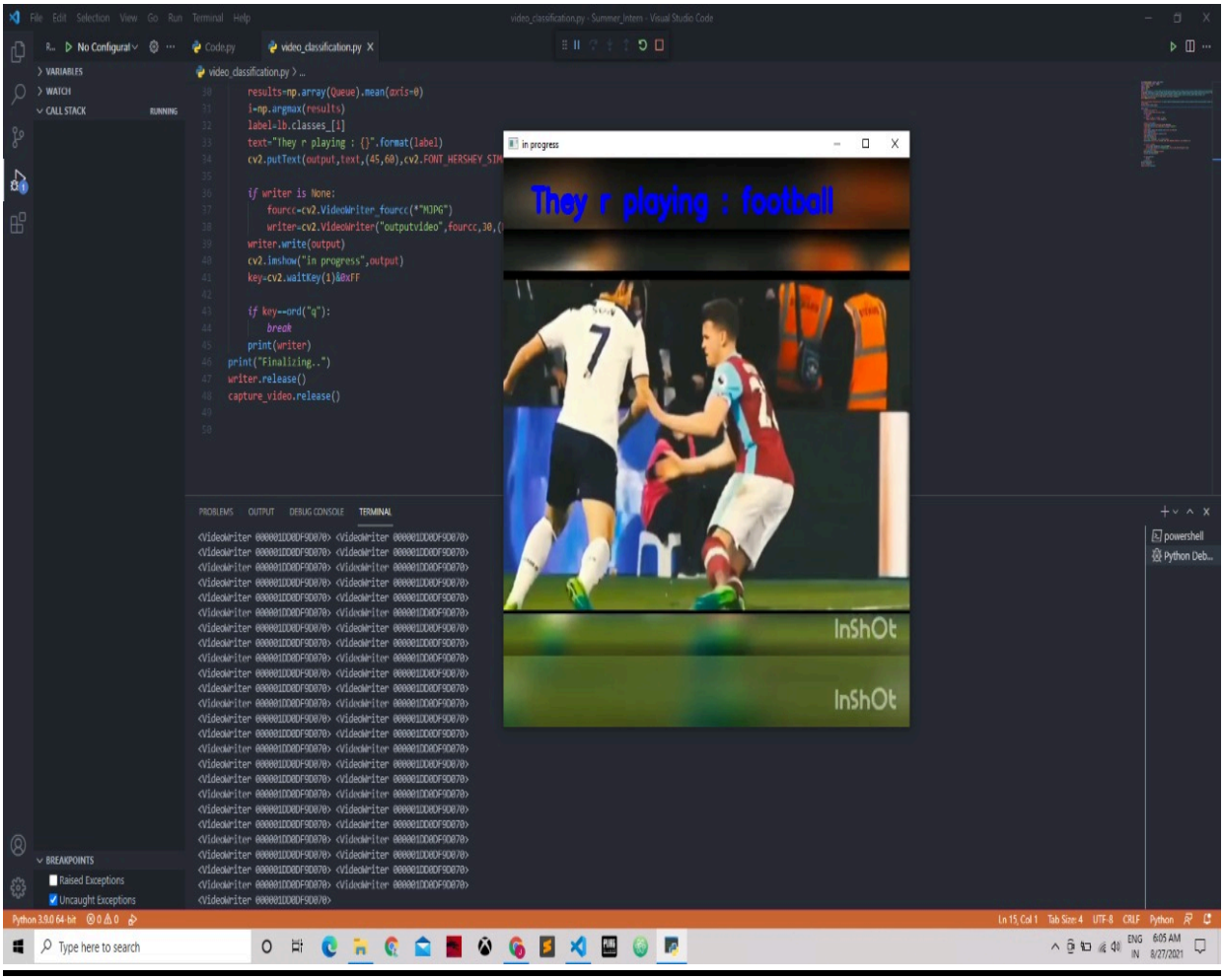
The model displays the activity name as wrestling.

OUTPUT OF THE PROJECT:

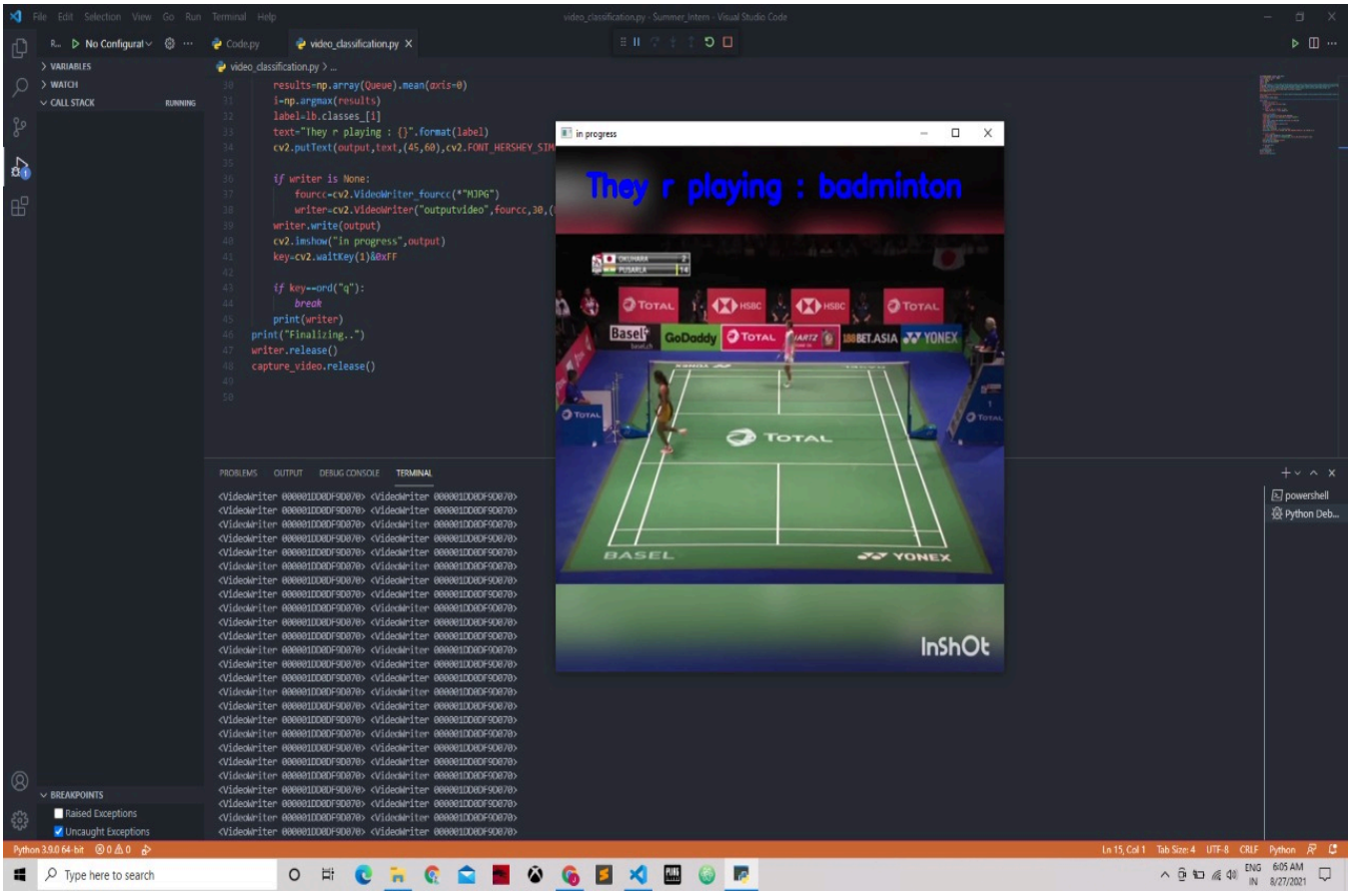
CASE 01:



CASE 02:


video_classification.py - Summer_Intern - Visual Studio Code
video_classification.py
results=np.array(Queue).mean(axis=0)
i=np.argmax(results)
label=lb.classes_[i]
text="They r playing : {}".format(label)
cv2.putText(output,text,(45,60),cv2.FONT_HERSHEY_STA
if writer is None:
fourcc=cv2.VideoWriter_fourcc(*"MJPG")
writer=cv2.VideoWriter("outputvideo",fourcc,30,(
writer.write(output)
cv2.imshow("in progress",output)
key=cv2.waitKey(1)&0xFF
if key==ord("q"):
break
print(writer)
print("Finalizing..")
writer.release()
capture_video.release()
in progress
They r playing : football
InShOt
InShOt
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
Python Deb...
Raised Exceptions
Uncaught Exceptions
Type here to search

CASE 03:


video_classification.py - Summer_Intern - Visual Studio Code
video_classification.py
results=np.array(Queue).mean(axis=0)
i=np.argmax(results)
label=lb.classes_[1]
text="They r playing : {}".format(label)
cv2.putText(output,text,(45,60),cv2.FONT_HERSHEY_SIM
if writer is None:
fourcc=cv2.VideoWriter_fourcc(*"MJPG")
writer=cv2.VideoWriter("outputvideo",fourcc,30,(
writer.write(output)
cv2.imshow("in progress",output)
key=cv2.waitKey(1)&0xFF
if key==ord("q"):
break
print(writer)
print("Finalizing..")
writer.release()
capture_video.release()
in progress
They r playing : badminton
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
Python Deb...
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
Type here to search

CONCLUSION

- At present and with the passage of time, the amount of sports training video data in the Internet is growing rapidly. In order to effectively manage and retrieve sports training video, accurate classification of sports training video is very important for consideration.

- Convolution neural network with deep learning is used for the classification purpose in the proposed model.

- After classification, event matching operation is performed, and video classification is realized according to similarity.

- The experimental results show that the proposed model can effectively determine all kinds of sports training videos and accurately detect the occurrence of events through convolution neural network(CNN), so as to achieve high-precision classification of sports training videos.

- Compared with other models, the proposed model has the advantages of simple implementation, fast processing speed, high classification accuracy, high generalization ability, and adaptability.

FUTURE SCOPE

- As many videos are present in the real world, an effective way to classify those videos is very important.

- In addition to that, we can improve this model further by the using different ways of classifying video based on video, audio and text.

- Video classification model can be used in real time situation like live streaming predictions.

References:

https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/

https://www.youtube.com/watch?v=BllyHlq5SsQ&list=PLxefhmF0pcPl_v-lLsqF3drP6NOoSYJR0

https://www.ijirase.com/assets/paper/issue_11/volume_4/V4-Issue-11-936-939.pdf

THANK YOU