Here is our azure code :

!pip install azure-cognitiveservices-vision-customvision

from azure.cognitiveservices.vision.customvision.training import CustomVisionTrainingClient

from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient

from azure.cognitiveservices.vision.customvision.training.models import ImageFileCreateBatch, ImageFileCreateEntry, Region

from msrest.authentication import ApiKeyCredentials

import time

#Big data set

project\_id = "b0ce3212-becb-44a2-818a-97fa5a666589"

ENDPOINT = "https://customtraining.cognitiveservices.azure.com/"

training\_key = "09369326f2ec40d2828f6d64613bc855"

prediction\_key = "03a5a51e760141049ce050958423226f"

prediction\_resource\_id = "/subscriptions/800e51fe-8ccb-4817-a0d3-3d80d2e88479/resourceGroups/mslearn-faceapi/providers/Microsoft.CognitiveServices/accounts/customTraining"

credentials = ApiKeyCredentials(in\_headers={"Training-key": training\_key})

trainer = CustomVisionTrainingClient(ENDPOINT, credentials)

prediction\_credentials = ApiKeyCredentials(in\_headers={"Prediction-key": prediction\_key})

predictor = CustomVisionPredictionClient(ENDPOINT, prediction\_credentials)

project = trainer.get\_project(project\_id, custom\_headers=None, raw=False)

project.id

iterations = trainer.get\_iterations(project\_id, custom\_headers=None, raw=False)

iterations[1].id

iteration = trainer.get\_iteration(project.id, iterations[1].id)

iteration.status

import requests

import datetime

import os

import math

from PIL import Image, ImageDraw

import cv2

import numpy as np

import glob

import pandas as pd

url = 'https://player.vimeo.com/play/2390529508?s=515388373\_1620000875\_11ee7012b5670addb0de5a96dd12f2ea&sid=22b955e28822e194396b81878f511f4218bd5ede1619990075&oauth2\_token\_id=&download=1'

r = requests.get(url, allow\_redirects=True)

open('video.mp4', 'wb').write(r.content)

def convertFpsToTime(Fps):

seconds = Fps /30

return str(datetime.timedelta(seconds=seconds))

!mkdir buffers #Folder to store input frames

!mkdir images #Folder to store output frames

#Function to process each input frame

def countImg(count):

save\_data = []

font = cv2.FONT\_HERSHEY\_SIMPLEX

with open("buffer" +str(count)+ ".jpg", mode="rb") as test\_data: #Importing input frame

results = predictor.detect\_image\_with\_no\_store(project.id, "Iteration5",test\_data) #Sending it to the model

image = cv2.imread("buffer" +str(count)+ ".jpg") #Reading the input frame

fish=0

if count<180or count>4470:

pred=0.7

else:

pred=0.6

for i in results.predictions:

if i.probability >pred:

#Calculating bounding box size

box = i.bounding\_box

h,w,\_ = image.shape

start\_point =( math.floor( box.left \* w) , math.floor(box.top \* h) )

end\_point = math.floor( (box.left \* w )+ box.width\* w) , math.floor( (box.top \* h) + box.height\* h )

color = (255, 0, 0)

thickness = 2

#Drawing bounding box

image = cv2.rectangle(image,start\_point,end\_point,color, thickness)

fish+=1

cv2.putText(image,'Count='+str(fish),(30,50), font, 2,(0,255,255),3)

cv2.imwrite("../images/" +str(count)+ ".jpg" ,image ) #Exporting output frame

return fish

cd buffers

FishNum=[] #List to store number of fish on each frame

cap = cv2.VideoCapture("../video.mp4") #Importing the video

# TRACKER INITIALIZATION

success, frame = cap.read()

count = 0

while(cap.isOpened()):

ret, img = cap.read()

if ret == False :

break

cv2.imwrite("buffer" +str(count)+ ".jpg" ,img ) #Storing input frame

print("read",count)

data = countImg(count) #Pass the frame to be processed

FishNum.append(data)

count +=1

#Exporting the images to create the final video

img\_array = []

scale\_percent = 60 # percent of original size

for i in range(count):

filename=f'images/{i}.jpg'

img = cv2.imread(filename)

width = int(img.shape[1] \* scale\_percent / 100)

height = int(img.shape[0] \* scale\_percent / 100)

dim = (width, height)

resized = cv2.resize(img, dim, interpolation = cv2.INTER\_AREA)

img\_array.append(resized)

out = cv2.VideoWriter('project2.mp4',cv2.VideoWriter\_fourcc(\*'DIVX'), 30, dim)

for i in range(len(img\_array)):

out.write(img\_array[i])

out.release()

!gsutil -q -m cp Data1.csv /content/drive/MyDrive/Azure1

!gsutil -q -m cp project.mp4 /content/drive/MyDrive/Azure1

OUTPUT = pd.DataFrame(FishNum)

OUTPUT.to\_csv('Data1.csv')

ret, img\_encode = cv2.imencode('.jpg', img)

str\_encode = img\_encode.tostring() #Convert array to binary type

f4 = BytesIO(str\_encode)

f5 = BufferedReader(f4) #Convert to \_io.BufferedReader type

And here is our plotting code

import random

import collections

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

def Average(lst):

Average = int(sum(lst) / len(lst))

return Average

def simpler(Fish\_Number\_List):

simp = []

x = 16

simp.append(Average(Fish\_Number\_List[:x]))

while 1:

try:

simp.append(Average(Fish\_Number\_List[x:x+30]))

x += 30

except:

if len(Fish\_Number\_List[x:]) > 0:

simp.append(Average(Fish\_Number\_List[x:]))

print(x)

break

return simp

def plot(x, y, color, label, order, max\_x, max\_y, linewidth=1, markersize=12):

markerfacecolor = color

plt.subplot(3, 1, order)

plt.plot(x, y, color=color, linewidth=linewidth, markerfacecolor=markerfacecolor, markersize=markersize,

label=label)

plt.grid()

plt.legend([label])

plt.ylim(0, max\_y + 10)

plt.xlim(0, (max\_x + 2) / 30)

plt.yticks(np.arange(0, max\_y + 10, 5))

plt.xticks(np.arange(0, max\_x + 2, 200) / 30)

plt.xlabel('Seconds')

plt.ylabel('Fish')

df = pd.read\_csv('Data1.csv')

Fish\_Number\_List = df['num'].tolist()

# Ordinary Average every 3 frames

maximum = []

time = []

frame = []

t = 0.0

MovAvgList = []

MovAvgFrameLst = []

for p in range(len(Fish\_Number\_List)):

if (p+2) == len(Fish\_Number\_List):

break

MovAvgFrameLst.append(Fish\_Number\_List[p])

MovAvgFrameLst.append(Fish\_Number\_List[p+1])

MovAvgFrameLst.append(Fish\_Number\_List[p+2])

MovAvgList.append(Average(MovAvgFrameLst))

temp\_lst = [Fish\_Number\_List[p], Fish\_Number\_List[p+1], Fish\_Number\_List[p+2]]

maximum.append(max(temp\_lst))

frame.append(t)

t += 1

MovAvgFrameLst.clear()

temp\_lst.clear()

frame.append(t + 0.1)

MovAvgList.append(Fish\_Number\_List[-2])

frame.append(t + 0.2)

MovAvgList.append(Fish\_Number\_List[-1])

for t in frame:

time.append(t/30)

simp = simpler(Fish\_Number\_List)

simp2 = simpler(MovAvgList)

simp3 = simpler(maximum)

max\_x = max([t])

max\_y = max([max(MovAvgList), max(Fish\_Number\_List), max(maximum)])

plot(time, Fish\_Number\_List, 'blue', 'Fish List', 1, max\_x, max\_y)

plot(time, MovAvgList, 'red', 'Moving Average', 2, max\_x, max\_y)

plot(time[:-2], maximum, 'orange', 'Moving Maximum', 3, max\_x, max\_y)

plt.figure()

plot(np.arange(0,len(simp), 1), simp, 'blue', 'Fish List', 1, max\_x, max\_y)

plot(np.arange(0,len(simp2), 1), simp2, 'red', 'Moving Average', 2, max\_x, max\_y)

plot(np.arange(0,len(simp3), 1), simp3, 'orange', 'Moving Maximum', 3, max\_x, max\_y)

plt.show()