```
# -*- coding: utf-8 -*-
"""Counting fish.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1C7InTGa9E3reLZK1nEycQ2UXmClgZFFX

**Initiating Azure**
"""

!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

"""**Model details**"""

#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) #get iteration data

iteration.status

import requests
import datetime
import os
import math
from PIL import Image, ImageDraw
import cv2
from io import BufferedReader, BytesIO
import numpy as np
import glob
import pandas as pd

"""**Fetching the Input Video**"""

url = 'https://player.vimeo.com/play/2390529508?s=515388373_1620000875_11ee7012b5670addb0de5a96dd12f2ea&sid=22b955e28822e194396b81878f511f4218bd5ede1619990075&oauth2_token_id
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 count the objects on each frame**"""

#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

!pwd

"""**Video Processing into frames**"""

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

cd ..

save_data=np.array(save_data)
print(save_data.shape[0])

"""**Converting output frames to  a video**"""

#Exporting the images to create the final video
img_array = []

scale_percent = 60 # percent of original size
for i in range(4544):
    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()

"""**Exporting the video to google drive**"""

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



"""Storing the data into a CSV file"""

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
```