import cv2

import numpy as np

import tensorflow as tf

model = tf.saved\_model.load('path\_to\_your\_saved\_model')

def detect\_objects(frame):

input\_tensor = tf.convert\_to\_tensor(frame)

input\_tensor = input\_tensor[tf.newaxis, ...]

detections = model(input\_tensor)

return detections

video\_path = 'path\_to\_your\_video\_file.mp4'

cap = cv2.VideoCapture(video\_path)

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

detections = detect\_objects(frame)

cv2.imshow('Object Detection', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

tracker = cv2.TrackerCSRT\_create()

def initialize\_tracker(frame, bbox):

tracker.init(frame, bbox)

def update\_tracker(frame):

ok, bbox = tracker.update(frame)

if ok:

p1 = (int(bbox[0]), int(bbox[1]))

p2 = (int(bbox[0] + bbox[2]), int(bbox[1] + bbox[3]))

cv2.rectangle(frame, p1, p2, (0, 255, 0), 2, 1)

else:

cv2.putText(frame, "Tracking failure detected", (100,80), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75,(0,0,255),2)

return frame

for detection in detections:

bbox = detection['bbox'] # Extract bounding box coordinates

initialize\_tracker(frame, bbox)

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

frame = update\_tracker(frame)

cv2.imshow('Object Tracking', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

img\_path = "images/blueangels.jpg"

prev\_time = time.time()

img = Image.open(img\_path)

detections = detect\_image(img)

inference\_time = datetime.timedelta(seconds=time.time() - prev\_time)

print ('Inference Time: %s' % (inference\_time))

cmap = plt.get\_cmap('tab20b')

colors = [cmap(i) for i in np.linspace(0, 1, 20)]

img = np.array(img)

plt.figure()

fig, ax = plt.subplots(1, figsize=(12,9))

ax.imshow(img)

pad\_x = max(img.shape[0] - img.shape[1], 0) \* (img\_size / max(img.shape))

pad\_y = max(img.shape[1] - img.shape[0], 0) \* (img\_size / max(img.shape))

unpad\_h = img\_size - pad\_y

unpad\_w = img\_size - pad\_x

if detections is not None:

unique\_labels = detections[:, -1].cpu().unique()

n\_cls\_preds = len(unique\_labels)

bbox\_colors = random.sample(colors, n\_cls\_preds)

for x1, y1, x2, y2, conf, cls\_conf, cls\_pred in detections:

box\_h = ((y2 - y1) / unpad\_h) \* img.shape[0]

box\_w = ((x2 - x1) / unpad\_w) \* img.shape[1]

y1 = ((y1 - pad\_y // 2) / unpad\_h) \* img.shape[0]

x1 = ((x1 - pad\_x // 2) / unpad\_w) \* img.shape[1]

color = bbox\_colors[int(np.where(

unique\_labels == int(cls\_pred))[0])]

bbox = patches.Rectangle((x1, y1), box\_w, box\_h,

linewidth=2, edgecolor=color, facecolor='none')

ax.add\_patch(bbox)

plt.text(x1, y1, s=classes[int(cls\_pred)],

color='white', verticalalignment='top',

bbox={'color': color, 'pad': 0})

plt.axis('off')

plt.savefig(img\_path.replace(".jpg", "-det.jpg"),

bbox\_inches='tight', pad\_inches=0.0)

plt.show()