TF & KERAS. РОБОТА із ЗОБРАЖЕННЯМИ

Файл: TF_KERAS_Image_05_002

Нейронна мережа YOLO. Інференс YOLOv3. Приклад детектування та локалізації об'єктів в відепотоці з камери

!!! УВАГА. Необхідно мати підключену відеокамеру

Базується на матеріалі наступних публікацій

- YOLOv3: An Incremental Improvement https://pjreddie.com/media/files/papers/YOLOv3.pdf
- OpenCV YOLO - object detection https://opencv-tutorial.readthedocs.io/en/latest/yolo/yolo.html
- YOLOv3 – Deep Learning Based Object Detection https://learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/

Загальні відомості YOLOv3 дивись

Використовується навчана YOLOv3 модель з бібліотеки OpenCV

Імпорт бібліотек

```
import cv2 as cv
import numpy as np
```

Завантаження імен класів та визначення випадкових кольорів

```
path = '... /'
classes = open(path + 'yolo/coco.names').read().strip().split('\n')
np.random.seed(42)
colors = np.random.randint(0, 255, size=(len(classes), 3), dtype='uint8')
```

Завантаження конфігурації моделі та її ваг та визначення обєкта 'net' (модель YOLO3)

```
net = cv.dnn.readNetFromDarknet(path + 'yolov3.cfg',
                                path + 'yolov3.weights')
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)
```

Функція отримення імен вихідних YOLO шарів

```
# def getOutputsNames(net):
    # Get the names of all the layers in the network
    layersNames = net.getLayerNames()
    # Get the names of the output layers,
    # i.e. the layers with unconnected outputs
    return [layersNames[i - 1] for i in net.getUnconnectedOutLayers()]#
```

Створення об'єкту захоплення кадру з відеопотоку камери

```
cap = cv.VideoCapture(0)
```

ЦИКЛ ЗЧИТУВАННЯ КАДРІВ та ФОРМУВАННЯ BOXів

```
while True:
    # Зчиутємо наступний кадр з відепотоку камери
    ret, frame = cap.read()
    if ret is False:
        break

    # формування блобу  з кадру
    blob = cv.dnn.blobFromImage(
        frame, 1/255.0, (416, 416), swapRB=True, crop=False)
    r = blob[0, 0, :, :]
    cv.imshow('BLOB', r)

    # Блоб до мережі
    net.setInput(blob)
    # Виходи мережі
    outputs = net.forward(getOutputsNames(net))

    boxes = []
    confidences = []
    classIDs = []
    h, w = frame.shape[:2]

    confThreshold = 0.5  # Зріз довіри
    nmsThreshold = 0.4   # Не максимальний поріг придушення

    # NMS – Non-Maximum Suppression
    for output in outputs:
        for detection in output:
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]
            if confidence > confThreshold:
                box = detection[:4] * np.array([w, h, w, h])
                (centerX, centerY, width, height) = box.astype("int")
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))
                box = [x, y, int(width), int(height)]
                boxes.append(box)
```

```
                confidences.append(float(confidence))
                classIDs.append(classID)

    indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)


    if len(indices) > 0:
        for i in indices.flatten():
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            color = [int(c) for c in colors[classIDs[i]]]
            cv.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            # print('Object Class "', classes[classIDs[i]], '"  '
            #      'Confidence = ', confidences[i])
            text = "{}: {:.4f}".format(classes[classIDs[i]], confidences[i])
            cv.putText(frame, text, (x, y - 5),
                       cv.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)

    cv.imshow('BLOB', frame)
    if cv.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

while True:
  # Зчиутємо наступний кадр з відепотоку камери
  ret, frame = cap.read()
  if ret is False:
    break

```
# формування блобу  з кадру
blob = cv.dnn.blobFromImage(
    frame, 1/255.0, (416, 416), swapRB=True, crop=False)
r = blob[0, 0, :, :]
cv.imshow('BLOB', r)

# Блоб до мережі
net.setInput(blob)
# Виходи мережі
outputs = net.forward(getOutputsNames(net))

boxes = []
confidences = []
classIDs = []
h, w = frame.shape[:2]

confThreshold = 0.5  # Зріз довіри
nmsThreshold = 0.4   # Не максимальний поріг придушення

# NMS – Non-Maximum Suppression
for output in outputs:
```

```
    for detection in output:
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]
        if confidence > confThreshold:
            box = detection[:4] * np.array([w, h, w, h])
            (centerX, centerY, width, height) = box.astype("int")
            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))
            box = [x, y, int(width), int(height)]
            boxes.append(box)
            confidences.append(float(confidence))
            classIDs.append(classID)

indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
```

```
if len(indices) > 0:
    for i in indices.flatten():
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])
        color = [int(c) for c in colors[classIDs[i]]]
        cv.rectangle(frame, (x, y), (x + w, y + h), color, 2)
        # print('Object Class "', classes[classIDs[i]], '"  '
        #       'Confidence = ', confidences[i])
        text = "{}: {:.4f}".format(classes[classIDs[i]], confidences[i])
        cv.putText(frame, text, (x, y - 5),
                   cv.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)
```

cap.release()
cv.destroyAllWindows()

```
```