**Mediapipe**

A MediaPipe Object Detector feladat lehetővé teszi több objektumosztály jelenlétének és helyének észlelését. Ezek az utasítások megmutatják, hogyan használhatja az Object Detector feladatot Pythonban.

Jellemzők

Bemeneti képfeldolgozás - A feldolgozás magában foglalja a kép elforgatását, átméretezését, normalizálását és színtérkonvertálását.

Címketérkép helymeghatározás - A megjelenített nevek nyelvének beállítása.

Pontszámküszöb - Az eredmények szűrése a predikciós pontszámok alapján.

Top-k felismerés - A számfelismerési eredmények szűrése.

Címkeengedélyezési lista és címkeelutasítási lista - Az észlelt kategóriák megadása.

Csomagok

Az Object Detector feladathoz a mediapipe pip csomag szükséges. A szükséges csomagokat a következő parancsokkal telepítheti:

$ python -m pip install mediapipe

| **Task inputs** | **Task outputs** |
| --- | --- |
| * Az objektumdetektor API a következő adattípusok egyikét fogadja el: * - Állóképek * - dekódolt videoképkockák * - Élő videótovábbítás | * Az objektumdetektor API a következőket adja ki * eredményeket az észlelt objektumokra vonatkozóan: * - Az objektum kategóriája * - Valószínűségi pontszám * - Határoló doboz koordinátái |

Import

Importálja a következő osztályokat az Object Detector feladat funkcióinak eléréséhez:

import mediapipe as mp  
from mediapipe.tasks import python  
from mediapipe.tasks.python import vision

Modell

Válasszon ki és töltsön le egy modellt, majd tárolja azt egy helyi könyvtárban:

model\_path = '/absolute/path/to/lite-model\_efficientdet\_lite0\_detection\_metadata\_1.tflite'

A BaseOptions objektum model\_asset\_path paraméterével adhatja meg a modell elérési útvonalát.

EfficientDet-Lite0 modell (ajánlott)

Az EfficientDet-Lite0 modell egy EfficientNet-Lite0 gerinchálózatot használ 320x320 bemeneti mérettel és BiFPN funkcióhálózattal. A modellt a COCO-adatkészlettel képeztük ki, amely egy nagyméretű tárgyfelismerő adathalmaz, amely 1,5 millió tárgypéldányt és 80 tárgycímkét tartalmaz. Tekintse meg a támogatott címkék teljes listáját. Az EfficientDet-Lite0 int8 modellként, float16 modellként vagy float32 modellként érhető el. Ez a modell ajánlott, mert egyensúlyt teremt a késleltetés és a pontosság között. Egyszerre pontos és elég könnyű számos felhasználási esethez.

EfficientDet-Lite2 modell

Az EfficientDet-Lite2 modell egy EfficientNet-Lite2 gerinchálózatot használ 448x448 bemeneti mérettel és BiFPN funkcióhálózattal. A modellt a COCO-adatkészlettel, egy nagyméretű tárgyfelismerési adatkészlettel képeztük, amely 1,5 millió tárgypéldányt és 80 tárgycímkét tartalmaz. Tekintse meg a támogatott címkék teljes listáját. Az EfficientDet-Lite2 int8 modellként, float16 modellként vagy float32 modellként érhető el. Ez a modell általában pontosabb, mint az EfficientDet-Lite0, de lassabb és memóriaigényesebb is. Ez a modell olyan felhasználási esetekben megfelelő, ahol a pontosság nagyobb prioritást élvez a sebességgel és a mérettel szemben.

SSD MobileNetV2 modell

Az SSD MobileNetV2 modell egy MobileNetV2 gerinchálózatot használ 256x256 bemeneti mérettel és SSD funkcióhálózattal. A modellt a COCO adatkészlettel képeztük ki, amely egy nagyméretű tárgyfelismerő adatkészlet, amely 1,5 millió tárgypéldányt és 80 tárgycímkét tartalmaz. Tekintse meg a támogatott címkék teljes listáját. Az SSD MobileNetV2 int8 modellként vagy float modellként érhető el. Ez a modell gyorsabb és könnyebb, mint az EfficientDet-Lite0, de általában kevésbé pontos. Ez a modell olyan felhasználási esetekre alkalmas, amelyek gyors, könnyű modellt igényelnek, amely némi pontosságot áldoz fel.

Modell referenciaértékek

Itt vannak a fenti, előre betanított modellek referenciaértékei. A késleltetési eredmény a Pixel 6 és a CPU 1 szálon készült.

| **Model Name** | **Model size** | **MAP (mean average precision)** | **CPU Latency** | **GPU Latency** |
| --- | --- | --- | --- | --- |
| EfficientDet-Lite0 float32 model | 14M | 24.38% | 51.11ms | 33.01ms |
|  |  |  |  |  |
| EfficientDet-Lite0 float16 model | 6.6M | 24.37% | 48.99ms | 33.13ms |
|  |  |  |  |  |
| EfficientDet-Lite0 int8 model | 4.4M | 24.08% | 27.8ms | - |
|  |  |  |  |  |
| EfficientDet-Lite2 float32 model | 21.7M | 31.70% | 158.51ms | 56.62ms |
|  |  |  |  |  |
| EfficientDet-Lite2 float16 model | 11M | 31.67% | 162.91ms | 57.00ms |
|  |  |  |  |  |
| EfficientDet-Lite2 int8 model | 7.2M | 31.21% | 82.45ms | - |
|  |  |  |  |  |
| SSD MobileNetV2 float model | 11M | 21.20% | 31.26ms | 24.55ms |
|  |  |  |  |  |
| SSD MobileNetV2 int8 model | 3.3M | 20.10% | 16.72ms | - |

Modellkövetelmények és metaadatok

Ez a szakasz az egyéni modellekkel szemben támasztott követelményeket ismerteti, ha úgy dönt, hogy modellt készít a feladathoz. Az egyéni modelleknek TensorFlow Lite formátumúaknak kell lenniük, és tartalmazniuk kell a modell működési paramétereit leíró metaadatokat.

#### Tervezési követelmények

| **Input** | **Shape** | **Description** |
| --- | --- | --- |
| Input image | Float32 tensor of shape[1, height, width, 3] | A normalizált bemeneti kép. |
| **Output** | **Shape** | **Description** |
| detection\_boxes | Float32 tensor of shape [1, num\_boxes, 4] | Az egyes észlelt objektumok dobozhelyzete. |
| detection\_classes | Float32 tensor of shape [1, num\_boxes] | Az egyes észlelt objektumok osztályneveinek indexei. |
| detection\_scores | float32 tensor of shape [1, num\_boxes] | Előrejelzési pontszámok minden egyes észlelt  objektumhoz. |
| num\_boxes | Float32 tensor of size 1 | Az észlelt dobozok száma. |

Metadata követelmények

| **Parameter** | **Description** | **Description** |
| --- | --- | --- |
| input\_norm\_mean | A bemeneti tenzor normalizálásához használt középérték. | A normalizált bemeneti kép. |
| input\_norm\_std | A bemeneti tenzor normalizálásához használt mező normája. | Az egyes észlelt objektumok dobozhelyzete. |
| label\_file\_paths | A kategória tenzorcímkefájlok elérési útvonalai. Ha a modellnek nincsenek címkefájljai, adjon meg egy üres listát. | Az egyes észlelt osztályok neveinek indexei  objektumhoz. |
| score\_calibration\_md | Az osztályozási tenzorban végzett pontszám-kalibrálási műveletre vonatkozó információk. Ez a paraméter nem szükséges, ha a modell nem használja a pontszámok  kalibrálást. | Előrejelzési pontszámok minden egyes észlelt objektumhoz. |
| num\_boxes | Float32 1 méretű tenzor | Az észlelt dobozok száma. |

A feladat létrehozása

A feladat létrehozásához használja a create\_from\_options függvényt. A create\_from\_options függvény olyan konfigurációs beállításokat fogad el, mint a futtatási mód, a megjelenített nevek helyi nyelve, az eredmények maximális száma, a megbízhatósági küszöb, a kategória engedélyezési listája és a tiltási lista. Ha nem állít be konfigurációs opciót, a feladat az alapértelmezett értéket használja.

Az Objektumdetektor feladat többféle bemeneti adattípust támogat: állóképek, videofájlok és élő videofolyamok.

import mediapipe as mp  
  
BaseOptions = mp.tasks.BaseOptions  
ObjectDetector = mp.tasks.vision.ObjectDetector  
ObjectDetectorOptions = mp.tasks.vision.ObjectDetectorOptions  
VisionRunningMode = mp.tasks.vision.RunningMode  
  
options = ObjectDetectorOptions(  
    base\_options=BaseOptions(model\_asset\_path='/path/to/model.tflite'),  
    max\_results=5,  
    running\_mode=VisionRunningMode.VIDEO)  
  
with ObjectDetector.create\_from\_options(options) as detector:  
  # The detector is initialized. Use it here.  
  # ...

Konfigurációs lehetőségek

Ez a feladat a következő konfigurációs lehetőségekkel rendelkezik Python alkalmazásokhoz:

| **Option Name** | **Description** | **Value Range** | **Default Value** |
| --- | --- | --- | --- |
| running\_mode | A feladat futási módjának beállítása. Az objektumdetektornak három üzemmódja van:  KÉP: Az objektumok felismerésének üzemmódja egyetlen kép bemenetén.  VIDEÓ: A videó dekódolt képkockáin lévő objektumok felismerésének üzemmódja.  LIVE\_STREAM: A bemeneti adatok élő adatfolyamán, például egy kamerából származó objektumok észlelésére szolgáló üzemmód. Ebben az üzemmódban az resultListener-t meg kell hívni egy olyan figyelő beállításához, amely aszinkron módon fogadja az észlelési eredményeket. | {IMAGE, VIDEO, LIVE\_STREAM} | IMAGE |
| display\_names | Beállítja a feladat modelljének metaadataiban megadott megjelenítési nevekhez használandó címkék nyelvét, ha rendelkezésre áll. Az alapértelmezett érték en az angol nyelvre. A TensorFlow Lite Metadata Writer API segítségével helyi címkéket adhat hozzá egy egyéni modell metaadataihoz. | Lokális kód | en |
| max\_results | Beállítja a legmagasabb pontszámú észlelési eredmények opcionális maximális számát. | Bármilyen pozitív szám | -1 (all results are returned) |
| score\_threshold | Beállítja az előrejelzési pontszám küszöbértékét, amely felülírja a modell metaadataiban megadott küszöbértéket (ha van ilyen). Az ezen érték alatti eredmények elutasításra kerülnek. | Bármilyen float | Not set |
| category\_allow\_list | Beállítja a megengedett kategórianevek opcionális listáját. Ha nem üres, akkor a rendszer kiszűri azokat az észlelési eredményeket, amelyek kategórianeve nem szerepel ebben a listában. A duplikált vagy ismeretlen kategórianevek figyelmen kívül maradnak. Ez az opció és a category\_deny\_list kölcsönösen kizárja egymást, és mindkettő használata hibát eredményez. | Bármilyen string | Not set |
| category\_deny\_list | Beállítja a nem engedélyezett kategórianevek opcionális listáját. Ha nem üres, akkor a rendszer kiszűri azokat az észlelési eredményeket, amelyek kategórianeve szerepel ebben a halmazban. A duplikált vagy ismeretlen kategórianevek figyelmen kívül maradnak. Ez az opció és a category\_allow\_list kölcsönösen kizárja egymást, és mindkettő használata hibát eredményez. | Bármilyen string | Not set |

Adatok előkészítése

Készítse elő a bemenetet képfájlként vagy numpy tömbként, majd konvertálja azt mediapipe.Image objektummá. Ha a bemenet egy videófájl vagy egy webkameráról származó élő stream, akkor egy külső könyvtárat, például az OpenCV-t használhatja a bemeneti képkockák numpy tömbként történő betöltéséhez.

A következő példák elmagyarázzák és bemutatják, hogyan kell előkészíteni az adatokat a feldolgozáshoz az egyes rendelkezésre álló adattípusok esetében:

import mediapipe as mp  
  
# Use OpenCV’s VideoCapture to load the input video.  
  
# Load the frame rate of the video using OpenCV’s CV\_CAP\_PROP\_FPS  
# You’ll need it to calculate the timestamp for each frame.  
  
# Loop through each frame in the video using VideoCapture#read()  
  
# Convert the frame received from OpenCV to a MediaPipe’s Image object.  
mp\_image = mp.Image(image\_format=mp.ImageFormat.SRGB, data=numpy\_frame\_from\_opencv)

A feladat futtatása

A következtetések kiváltásához meghívhatja a detektáló függvények egyikét. Az Object Detector feladat visszaadja a bemeneti képen vagy képkockán belül észlelt objektumokat.

# Calculate the timestamp of the current frame  
frame\_timestamp\_ms = 1000 \* frame\_index / video\_file\_fps  
  
# Perform object detection on the video frame.  
detection\_result = detector.detect\_for\_video(mp\_image, frame\_timestamp\_ms)

Vegye figyelembe a következőket:

Ha videó vagy élő stream üzemmódban fut, akkor a bemeneti képkocka időbélyegzőjét is meg kell adnia az Objektumdetektor feladatnak.

A kép- vagy videomodellben történő futtatáskor az Object Detector task blokkolja az aktuális szálat, amíg az be nem fejezi a bemeneti kép vagy képkocka feldolgozását.

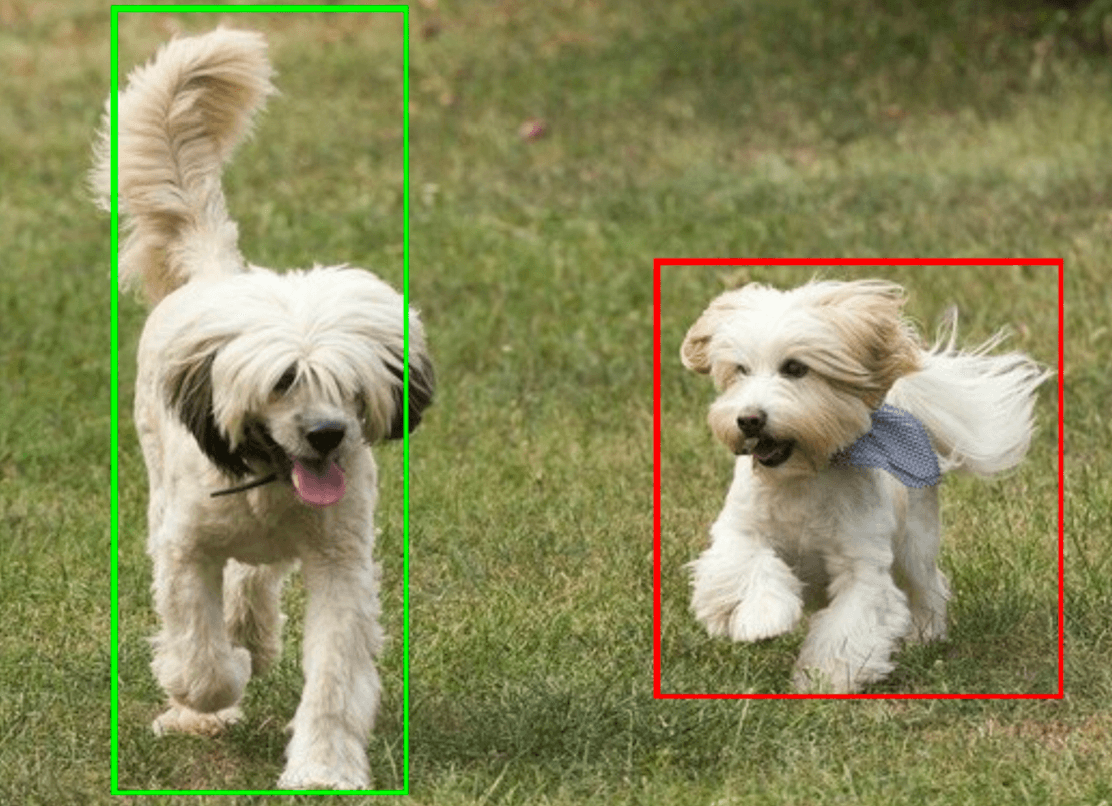
Amikor az objektumdetektor feladat élő stream üzemmódban fut, az objektumdetektor feladat nem blokkolja az aktuális szálat, hanem azonnal visszatér. Minden alkalommal, amikor befejezte egy bemeneti képkocka feldolgozását, meghívja az eredményhallgatóját az észlelés eredményével. Ha a detektáló függvényt akkor hívja meg, amikor az Object Detector task éppen egy másik képkocka feldolgozásával van elfoglalva, az új bemeneti képkockát figyelmen kívül hagyja.

Az eredmények kezelése és megjelenítése

A következtetés lefutása után az Object Detector feladat egy ObjectDetectionResult objektumot ad vissza, amely leírja a bemeneti képen talált objektumokat.

Az alábbiakban egy példa látható a feladat kimeneti adataira:

ObjectDetectionResult:  
 Detection #0:  
  Box: (x: 355, y: 133, w: 190, h: 206)  
  Categories:  
   index       : 17  
   score       : 0.73828  
   class name  : dog  
 Detection #1:  
  Box: (x: 103, y: 15, w: 138, h: 369)  
  Categories:  
   index       : 17  
   score       : 0.73047  
   class name  : dog

A következő kép a feladat kimenetének vizualizációját mutatja:

Forrás

MediaPipe-Object detection/python (<https://developers.google.com/mediapipe/solutions/vision/object_detector/python#video>; https://developers.google.com/mediapipe/solutions/vision/object\_detector/index)