UASLP
Universidad Autónoma de San Luis Potosí

FACULTAD DE INGENIERÍA
Área de Ciencias de la Computación

Object Detection in Images using OpenCV DNN

Tercer Examen Parcial

Visión Computacional

Nombre: Rodriguez Donjuan Giovana Guadalupe

Clave Única:A339987

Profesor: Cesar Augusto Puente Montejano

Fecha de entrega: 24 de mayo de 2024

Planteamiento del problema a resolver

Implementar un detector de objetos utilizando el módulo DNN de OpenCV.

a. Investigar sobre el dataset de Imágenes de objetos MS COCO. Decir quién y cuándo lo creó. Qué contiene, cuantas clases y número de imágenes. Agregar para que tipo de problemas se ha utilizado y el enlace de donde se puede bajar públicamente. Finalmente, mencione otros 3 datasets utilizados popularmente para la detección de objetos.
b. Utilizar la biblioteca OpenCV para implementar un programa que detecte objetos que pertenezcan a alguna(s) de clase(s) del dataset MS COCO. Para ello puede basarse en el ejercicio que vimos en clase. Ver siguiente enlace: https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive guide/
El ejercicio comienza bajo el título que dice: Object Detection in Images using OpenCV DNN. Para el programa que usted realice, puede utilizar cualquiera de las DNN y frameworks soportados que vienen en la documentación de OpenCV. Pero para obtener un buen desempeño, debería asegurarse que la DNN que elija está ya pre-entrenada con el dataset MS COCO.
NOTA IMPORTANTE: El modelo utilizado deber ser diferente al que se usa en el ejemplo: MobileNet SSD. (NO USAR ESTE) En cualquier caso, deberá también entregar una breve semblanza de la DNN que eligió para su trabajo (nombre de la DNN, quien la propuso, descripción breve de la arquitectura, y para qué problema se propuso).
c. Para probar su implementación, debe tomar una foto (o video, si su capacidad de cómputo lo permite) usted mismo donde aparezcan objetos de las clases de MS COCO. Si es muy difícil hacer una foto o video con las clases que usted eligió para probar, puede utilizar alguna imagen o video del dominio público y poner la referencia de donde lo obtuvo. El caso es que NO utilice fotos o videos del dataset MS COCO.

Descripción de la solución

MS COCO

El conjunto de datos COCO (Common Objects in Context) es un conjunto de datos de reconocimiento de imágenes a gran escala para tareas de detección, segmentación y subtítulos de objetos publicado por Microsoft en el año 2014,sus principales autores son Tsung-Yi Lin Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick James Hays, Pietro Perona Deva, Ramanan C. Lawrence Zitnick, Piotr Dollar.
El dataset de MS COCO contiene 91 categorías de objetos comunes como materiales y objetos sin límites claros (cielo, calle, césped, etc.) que proporcionan información contextual significativa. Además 80 categorías incluyen "cosas" para las que las instancias individuales pueden etiquetarse fácilmente (persona, coche, silla, etc.) y 82 categorías que tienen más de 5.000 instancias etiquetadas. En total, el conjunto de datos tiene 2.500.000 instancias etiquetadas en 328.000 imágenes. Hay 250.000 personas con 17 puntos clave diferentes, popularmente utilizados para la estimación de poses,algunos de los puntos clave son nariz,ojos,orejas,rodillas,etc.

Dataset MS COCO: https://cocodataset.org/#download

DATASET usandos para la detección de objetos

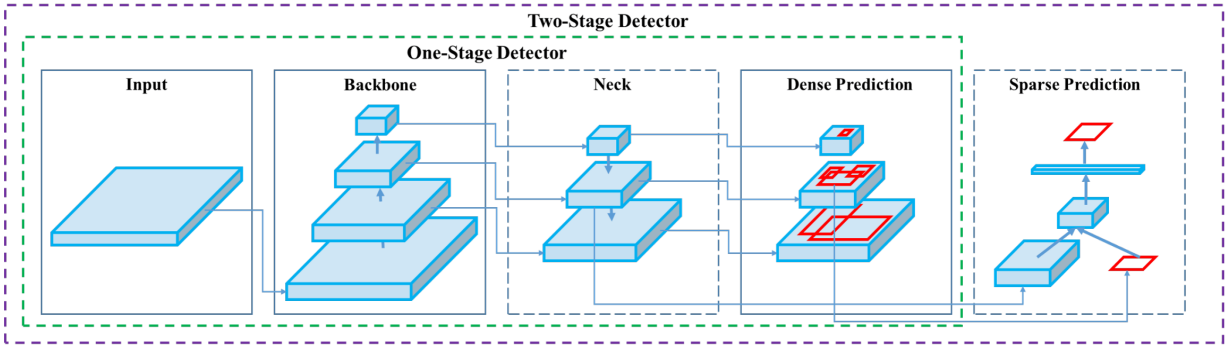
DOTA v2.0: DOTA es un conjunto de datos a gran escala para la detección de objetos en imágenes aéreas. Se puede utilizar para desarrollar y evaluar detectores de objetos en imágenes aéreas.
*NuScenes:*El dataset de nuScenes es un conjunto de datos público a gran escala para la conducción autónoma desarrollado por el equipo de Motional (anteriormente nuTonomy).
PASCAL VOC: El conjunto de datos PASCAL Visual Object Classes (VOC) 2012 contiene 20 categorías de objetos. Este conjunto de datos se ha utilizado ampliamente como punto de referencia para tareas de detección de objetos, segmentación semántica y clasificación. El conjunto de datos PASCAL VOC se divide en tres subconjuntos: 1.464 imágenes para entrenamiento, 1.449 imágenes para validación y un conjunto de pruebas privado.

YOLOv4

YOLOv4(You Only Look Once, es un modelo de detección de objetos en tiempo real lanzado en 2020 por Alexey Bochkovskiy, Chien-Yao Wang y Hong-Yuan Mark Liao.Desarrollado para abordar las limitaciones de versiones anteriores de YOLO como YOLOv3 y otros modelos de detección de objetos. A diferencia de otros detectores de objetos basados en

redes neuronales convolucionales (CNN), YOLOv4 no sólo es aplicable a los sistemas de recomendación, sino también a la gestión autónoma de procesos y a la reducción de entradas humanas.
Arquitectura
YOLOv4 utiliza varias funciones innovadoras que trabajan juntas para optimizar su rendimiento. Entre ellas se encuentran las conexiones residuales ponderadas (WRC), las conexiones parciales en etapas cruzadas (CSP), la normalización de mini lotes cruzados (CmBN), el entrenamiento autoadversarial (SAT), la activación errónea, el aumento de datos en mosaico, la regularización DropBlock y la pérdida CIoU.

YOLOv4 se compone de varias partes: la entrada, la columna vertebral, el cuello y la cabeza. La columna vertebral de YOLOv4 está preentrenada en ImageNet y se utiliza para predecir las clases y los cuadros delimitadores de los objetos. La columna vertebral puede proceder de varios modelos, como VGG, ResNet, ResNeXt o DenseNet. La parte del cuello del detector se utiliza para recoger mapas de características de distintas etapas y suele incluir varias trayectorias ascendentes y varias descendentes. La parte de la cabeza es la que se utiliza para hacer las detecciones y clasificaciones finales de los objetos.



Código fuente: https://github.com/AlexeyAB/darknet

Funciones de OpenCV

cv.dnn.readNetFromDarknet()

Esta función carga el modelo, en este caso es YOLOv4, desde los archivos de configuración y pesos.
Parametros
configPath: Ruta al archivo de configuración .cfg del modelo.
weightsPath: Ruta al archivo de pesos .weights del modelo.

'net = cv2.dnn.readNetFromDarknet("DNN/yolov4-tiny.cfg", "DNN/yolov4-tiny.weights")'

cv.dnn.blobFromImage()

Esta función convierte una imagen en un blob, que es el formato de entrada requerido por el modelo DNN.
Parametros
image: La imagen de entrada.
scalefactor: Factor para escalar los valores de píxeles. Normaliza los píxeles a un rango de [0, 1].
size: Tamaño al que se redimensiona la imagen en pixeles.
mean: Valores para restar del promedio en cada canal. Si se usa (0, 0, 0), significa que no se resta nada.
swapRB: Si es True, intercambia el canal rojo con el canal azul, ya que los modelos de DNN usualmente usan RGB.
crop: Si es True, recorta la imagen después del redimensionamiento. Si se establece como False porque no se recorta nada.

'blob = cv2.dnn.blobFromImage(img, 1/255.0, (416, 416), (0, 0, 0), swapRB=True, crop=False)'

cv.dnn.NMSBoxes()

Esta función aplica NMS para filtrar las detecciones redundantes. La supresión no máxima (NMS) es una clase de algoritmos para seleccionar una entidad como lo pueden ser cuadros delimitadores, de entre muchas entidades superpuestas.

Parametros

boxes: Lista de cajas delimitadoras (bounding boxes) propuestas por el modelo.

scores: Lista de puntuaciones de confianza para cada caja.

score_threshold: Umbral para filtrar las cajas con baja puntuación de confianza. Las cajas con puntuación por debajo de este valor se descartan.

nms_threshold: Umbral de NMS. Las cajas con una superposición (IoU) mayor que este valor se suprimen.

'indexes = cv2.dnn.NMSBoxes(boxes, confidences, score_threshold=0.5, nms_threshold=0.4)'

net.setInput()

Establece el blob preprocesado como la entrada de la red.

Parametros

blob: El blob generado por cv.dnn.blobFromImage.

'net.setInput(blob)'

net.getUnconnectedOutLayersNames()

Obtiene los nombres de las capas de salida que no están conectadas a otras capas. Estas son las capas de salida finales del modelo, donde se obtienen las predicciones.

Parametros

last_layer: Nombres de las capas de salida no conectadas.

'last_layer = net.getUnconnectedOutLayersNames()'

net.forward()

Realiza la inferencia de avance (forward pass) a través de la red para obtener las predicciones.

Parametros

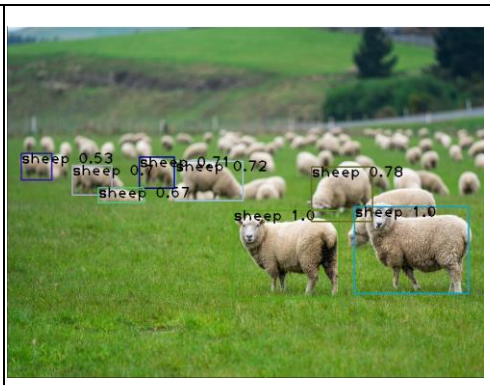
outputLayerNames: Nombres de las capas de salida a las que se desea obtener los resultados.

'layer_out = net.forward(last_layer)'

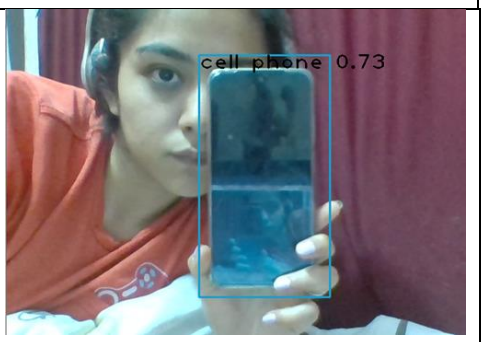
Descripción de los resultados

Como se puede observar tanto en la detección de imagen como en la detección a través de video, los resultados obtenidos son favorables y se detectan los objetos esperados. Las imágenes y objetos los elegí en base a la lista de objetos del dataset MSCOCO, además intenté que en una imagen se encontraran varios objetos para que se identificara más de uno y probar su eficiencia.
Los resultados y precisión de la captura de video son un poco menos precisos, esto lo atribuyo a la calidad de la cámara e iluminación.

DETECCIÓN POR FOTOS



DETECCIÓN POR VIDEO



Discusión

Decidi usar YOLOv4-Tiny porque en base a la investigación que realice para seleccionar una DNN, encontre que en comparación con otros modelos,YOLOv4-Tiny requiere menos procesamiento y memoria,lo que hace que sea más adecuado para dispositivos con recursos limitados como celulares,drones o cámaras de seguridad.Además de que su entrenamiento es más rapido.Aunque también es menos preciso que sus sucesores,como YOLOv8, que incluso puede dlasificar objetos dandole como entrada un video en lugar de una imagen.

Conclusión

Realizar este detector de imágenes fue una excelente asignación para poder entender a profundidad la detección de objetos,el proceso que se debe seguir,como lo es el cargar el modelo,guardar los nombres de los objetos,configurar la red ,etc.
Considero que OpenCV es una muy buena herramienta para la clasificación de objetos, ya que,su integración directa de DNN's facilita el uso de los modelos por sus funciones implementadas para el manejo.
Además de que no solo se limita a un modelo y esto hace que lo podamos probar con diferentes arquitecturas. Para terminar la función 'cv.dnn.NMSBoxes' hace un filtro de las secciones redundantes, obteniendo como salida resultados más precisos y ahorrando recursos.

Referencias

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2014). Microsoft COCO: Common objects in context. In *arXiv [cs.CV]*. http://arxiv.org/abs/1405.0312

Meel, V. (2023, October 1). *What is the COCO Dataset? What you need to know in 2024*. Viso.Ai. https://viso.ai/computer-vision/coco-dataset/

The PASCAL visual object classes homepage. (n.d.). Ox.ac.uk. Retrieved May 24, 2024, from http://host.robots.ox.ac.uk/pascal/VOC/

No title. (n.d.). Nuscenes.org. Retrieved May 24, 2024, from https://www.nuscenes.org/nuscenes

DOTA. (n.d.). Github.Io. Retrieved May 24, 2024, from https://captain-whu.github.io/DOTA/dataset.html

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. In *arXiv [cs.CV]*. http://arxiv.org/abs/2004.10934

Ultralytics. (2023, November 12). *YOLOv4*. Ultralytics.com. https://docs.ultralytics.com/es/models/yolov4/