CEIA - UBA - VISION POR COMPUTADORA 2

TRABAJO PRÁCTICO INTEGRADOR

DETECCIÓN DE PERSONAS Y AUTOELEVADORES

DOCENTES:

- Juan Ignacio Cornet
- Juan Ignacio Cavalieri
- Seyed Pakdaman

ALUMNOS:

- Sergio Hinojosa
- Leandro Albachiaro

DESCRIPCIÓN DEL PROBLEMA

En el entorno industrial la seguridad y eficiencia son fundamentales, por eso, un modelo utilizado para detectar personas y autoelevadores mediante modelos avanzados de Machine Learning puede ser de gran utilidad para prevenir accidentes, mejorar la planificación logística y garantizar un entorno industrial más seguro.

El dataset utilizado para entrenar el modelo se encuentra en roboflow: https://universe.roboflow.com/mohamed-traore-2ekkp/forklift-dsitv/dataset/2

CONJUNTO DE DATOS

Las características del dataset son:

- Resolución: 416x416px
- Cant. de Imágenes: 421
- División del dataset: 70%/20%/10%
- Cantidad de clases: 2 (Forklift, Person)

SOLUCIÓN PROPUESTA

Se entrenará un detector de objetos que permita encontrar la posición exacta de cada clase en cada una de las imágenes, siendo posible la existencia de más de una instancia en la misma foto y siendo capaz de diferenciarlas entre sí.

Se utilizarán la métrica mAP 50.

El Modelo deberá graficar en cada imagen un bounding box en cada instancia detectada, junto con su probabilidad correspondiente.

AUGMENTACIONES SUGERIDAS

- HorizontalFlip
- Rotate(limit=(-20, 20), p=0.5)
- RandomResizedCrop(height=size, width=size, scale=(0.7, 1.0), ratio=(0.9, 1.11), p=0.5)
- Blur(blur_limit=(2, 3), p=0.2)
- MedianBlur(blur_limit=(3, 5), p=0.2)
- CLAHE(clip_limit=(1, 4.0), tile_grid_size=(8, 8), p=0.3)
- RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=0.5)
- ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.09, p=0.4)

Las augmentaciones sugeridas se utilizan para ayudar al modelo a generalizar y reducir los problemas de iluminación y desenfoque de las imágenes, los cuales suelen ser comunes en los ambientes industriales.

MODELOS A ENTRENAR

- YOLO v5
- RETINANET

Se entrenaron distintos modelos de cada arquitectura, variando los parámetros y augmentaciones en cada uno.

YOLO v5

Se entrenaron 4 modelos de YOLO en total, utilizando los pesos pre-entrenados de “yolov5s” y “yolov5m” y utilizando o no las augmentaciones sugeridas.

Estos pesos fueron entrenados utilizando el dataset COCO.
https://cocodataset.org/#home

Link al repositorio de YOLO v5
https://github.com/ultralytics/yolov5

Pretrained Checkpoints

Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

RESULTADOS DE YOLO

Los modelos exhibieron una notable consistencia en sus rendimientos, siendo especialmente efectivos con la clase mayoritaria.

Aumentar el tamaño del dataset es clave para mejorar el funcionamiento del modelo.

Clase	YOLO v5m C/DA	YOLO v5m S/DA	YOLO v5s C/DA	YOLO v5s S/DA
Autoelevador	0.92	0.938	0.91	0.876
Persona	0.692	0.7	0.717	0.72
Total	0.806	0.819	0.813	0.798

RETINANET

Retinanet es usado ampliamente para la detección de objetos y es una poderosa alternativa a Yolo, SSD y Faster R-CNN.

La arquitectura de Retinanet se sustenta en Feature Pyramid Network (FPN) y en Focal Loss como función de perdida.

ENTRENAMIENTO

- Entrenamientos de 150, 300 y 500 épocas

- Sin pesos preentrenados
  - Optimizador: Stochastic Gradient Descent
    - mAP 0.50 = 0.26
  - Optimizador: Adam
    - mAP 0.50 = 0.46

ENTRENAMIENTO

Con pesos preentrenados. Basados en dataset de imageNet
(download.pytorch.org/models/resnet50-19c8e357.pth)

(catalog.ngc.nvidia.com/orgs/nvidia/models/resnet50_pyt_amp)

- Optimizador: Stochastic Gradient Descent
  - mAP 0.50 = 0.48
- Optimizador: Adam
  - mAP 0.50 = 0.493

ENTRENAMIENTO SIN PREENTRENO

Con el optimizador Adam se consiguio mejor performance, aunque el costo en tiempo de procesamiento no fue significativo respecto al optimizador SGD.

El performace de la metrica en general no fue bueno.

ENTRENAMIENTO CON PREENTRENO

Se esperaba que utilizando transfer learning se lograra acelerar el entrenamiento pudiendo entrenar más epocas en un tiempo similar, se logró una mejora en el rendimiento pero faltan ajustes para conseguir el rendimiento que se consiguió con YOLO.

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cache/torch/hub/c
100%|██████████| 97.8M/97.8M [00:00<00:00, 196MB/s]
BACKBONE     : resnet50
INFO:retinanet.models:BACKBONE     : resnet50
INPUT_PARAMS : MAX_SIZE=416, MIN_SIZE=416
INFO:retinanet.models:INPUT_PARAMS : MAX_SIZE=416, MIN_SIZE=416
NUM_CLASSES  : 2
INFO:retinanet.models:NUM_CLASSES  : 2

  | Name | Type      | Params
-----------------------------------
0 | net  | Retinanet | 36 M
INFO:retinanet.models:
  | Name | Type      | Params
-----------------------------------
0 | net  | Retinanet | 36 M
Epoch 96: 0%                    0/13 [00:00<?, ?it/s, loss=0.733, v_num=0, classification_loss=0.42, regression_loss=0.265, val_loss=0.85]
```

Loss por epoch

Colab: https://colab.research.google.com/drive/1LtXqNfGDjKyEEL7JQfgprDNJ1KKUCwqM#scrollTo=2faTFGtOEVj8