Entrega 5

DL

Entrena un modelo de deep learning con tus propias imágenes. Puedes utilizar la herramienta ultralytics y apoyarte en el ejemplo code/DL/yolotrain. Otra posibilidad es reproducir el ejemplo code/DL/UNET.

Para realizar este ejercicio, he tomado como referencia el directorio `code/DL/yolotrain` donde se ejecuta el fichero `facelabel.py` para obtener imágenes de referencia que reconocen la boca de una persona.

Estas se utilizan para entrenar un modelo de `yolo` ejecutando el comando `yolo detect train data=boca.yaml model=yolo11n.pt epochs=200 imgsz=640 augment=True` .

Una vez entrenado el modelo, se ejecuta el programa `yolo_run.py` para que reconozca dicha boca.

Para ello, he modificado el fichero `facelabel.py` para que reconozca los ojos de una persona, cambiando las coordenadas de la malla de la cara, obtenidas mediante `mp_face_mesh.FaceMesh()` .

Una vez modificado y comprobado el código, un ejemplo de ejecución es el siguiente:

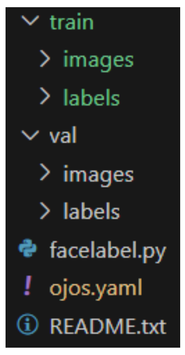
Tras ello, he entrenado el modelo de yolo, para lo cual he necesitado varias cosas.

- Tener definido un fichero `ojos.yaml` con el siguiente contenido:

```yaml
# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1,
path: C:\Users\jormo\Desktop\jorge\Universidad\4o\2oCuatri\VIA\DoscModificados\Entregas\Entrega5\src
train: train
val:   val
test:  train

names:
    0: ojos
```

- Para cumplir con el contenido del fichero anterior, he tenido que crear previamente los directorios `train` y `val` para guardar las imágenes producidas por el fichero `facelabel.py` , cuyo contenido es, en ambos casos, dos directorios con los nombres `images` y `labels` . La jerarquía de ficheros queda con la siguiente estructura:



- Esto es necesario para contener ciertas imágenes para el entrenamiento del modelo, y otras para su validación.
- Tras ello, al estar en windows, he necesitado ejecutar el siguiente comando para no tener errores con las librerías duplicadas. `set MP_DUPLICATE_LIB_OK=TRUE`

Una vez cumplidos estos requisitos, he ejecutado el comando `yolo detect train data=ojos.yaml model=yolo11n.pt epochs=200 imgsz=640 augment=True` para comenzar el entrenamiento del modelo, obteniendo como salida lo siguiente:

```
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs\detect\train
Starting training for 200 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/200         0G      5.135      8.292      3.584         16        640: 100%|██████████| 1/1 [00:05<00:00,  5.73s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,
                   all          1          1          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      2/200         0G      4.748      7.079      3.688         16        640: 100%|██████████| 1/1 [00:05<00:00,  5.09s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,
                   all          1          1          0          0          0          0

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
  0%|          | 0/1 [00:00<?, ?it/s]
```

Esto hará 200 iteraciones para entrenar el modelo, aunque en la imagen solo se aprecian 3.

Tras entrenar el modelo, se ha generado el directorio runs, el cual contiene, entre otras cosas, el mejor modelo obtenido del entrenamiento.

Este modelo llamado `best.pt` , es usado en el fichero `yolo_run.py` para reconocer los ojos de una persona.

El resultado de esa ejecución es el siguiente, donde como podemos comprobar, es el esperado: