**UNI**

**Maestría en inteligencia artificial 2024-I**

**MIA-07 Redes Neuronales y Aprendizaje Profundo**

**Grupo 4:**

* Acevedo Martínez, Ruddy.
* Fuertes Malca, Ayrton.
* Solís Almerco, Ricardo.
* Vilcahuamán Dolorier, Dennis.
* Zambrano Soto, José.

**Objetivo:**

Realizar la detección de imágenes en un partido de fútbol utilizando la arquitectura YOLOv7.

**Introducción:**

En este informe se documenta el proceso de entrenamiento y evaluación de un modelo de detección de objetos utilizando la arquitectura YOLOv7. El objetivo del proyecto fue identificar jugadores de fútbol, arqueros, árbitros y balones en imágenes de partidos de fútbol. Para lograr esto, se utilizó un conjunto de datos específico y se ajustaron diversos hiper-parámetros durante el entrenamiento del modelo.

**Pasos para Entrenar el Modelo:**

1. **Selección de la Arquitectura:**

Se seleccionó la arquitectura YOLOv7 debido a su eficiencia y precisión en tareas de detección de objetos. Se reutilizaron los pesos pre-entrenados obtenidos del dataset COCO (train2017, test2017, val2017).

El dataset COCO se puede obtener ejecutando el siguiente comando git bash: bash ./scripts/get\_coco.sh en la ruta del proyecto.

1. **Obtención del Dataset de Jugadores de Fútbol:**

El dataset específico de jugadores de fútbol se obtuvo de Roboflow: (https://universe.roboflow.com/roboflow-jvuqo/football-players-detection-3zvbc/dataset/12/download/yolov7pytorch?formCode=MG0AV3) Este dataset contiene cuatro categorías: 'ball', 'goalkeeper', 'player', 'referee'.

1. **Configuración del Entrenamiento:**

El entrenamiento del modelo se realizó utilizando el siguiente comando:

python train.py --weights yolov7.pt --data football-players/data.yaml --cfg cfg/training/yolov7.yaml --batch-size 8 --epochs 100 --img-size 640 640 --name football\_detection --hyp data/hyp.scratch.custom.yaml --workers 8 --device 0

1. **Hiper-parámetros Utilizados:**

Se ajustaron los siguientes hiper-parámetros para optimizar el entrenamiento:

lr0: 0.01 # initial learning rate (SGD=1E-2, Adam=1E-3)

lrf: 0.1 # final OneCycleLR learning rate (lr0 \* lrf)

momentum: 0.937 # SGD momentum/Adam beta1

weight\_decay: 0.0005 # optimizer weight decay 5e-4

warmup\_epochs: 3.0 # warmup epochs (fractions ok)

warmup\_momentum: 0.8 # warmup initial momentum

warmup\_bias\_lr: 0.1 # warmup initial bias lr

box: 0.05 # box loss gain

cls: 0.3 # cls loss gain

cls\_pw: 1.0 # cls BCELoss positive\_weight

obj: 0.7 # obj loss gain (scale with pixels)

obj\_pw: 1.0 # obj BCELoss positive\_weight

iou\_t: 0.20 # IoU training threshold

anchor\_t: 4.0 # anchor-multiple threshold

fl\_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)

hsv\_h: 0.015 # image HSV-Hue augmentation (fraction)

hsv\_s: 0.7 # image HSV-Saturation augmentation (fraction)

hsv\_v: 0.4 # image HSV-Value augmentation (fraction)

degrees: 0.0 # image rotation (+/- deg)

translate: 0.2 # image translation (+/- fraction)

scale: 0.9 # image scale (+/- gain)

shear: 0.0 # image shear (+/- deg)

perspective: 0.0 # image perspective (+/- fraction), range 0-0.001

flipud: 0.0 # image flip up-down (probability)

fliplr: 0.5 # image flip left-right (probability)

mosaic: 1.0 # image mosaic (probability)

mixup: 0.15 # image mixup (probability)

copy\_paste: 0.0 # image copy paste (probability)

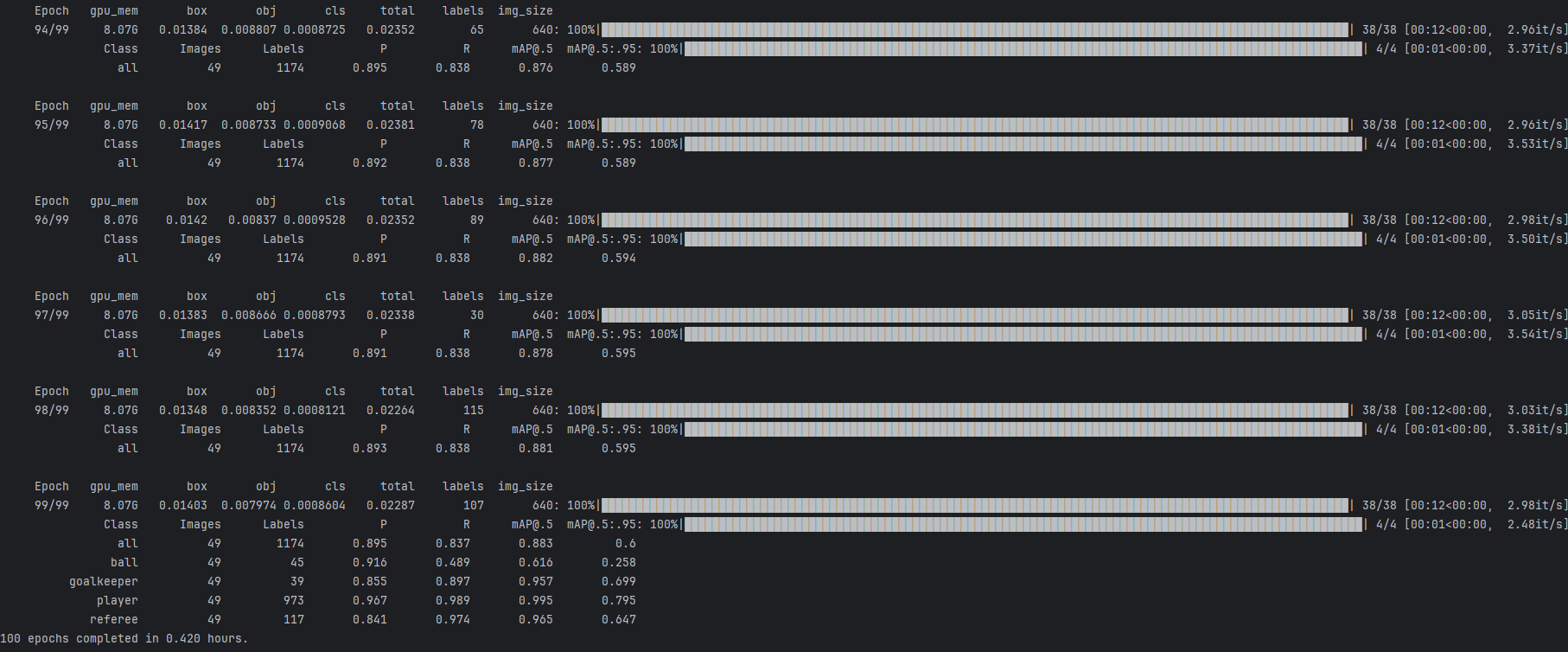
paste\_in: 0.15 # image copy paste (probability), use 0 for faster training

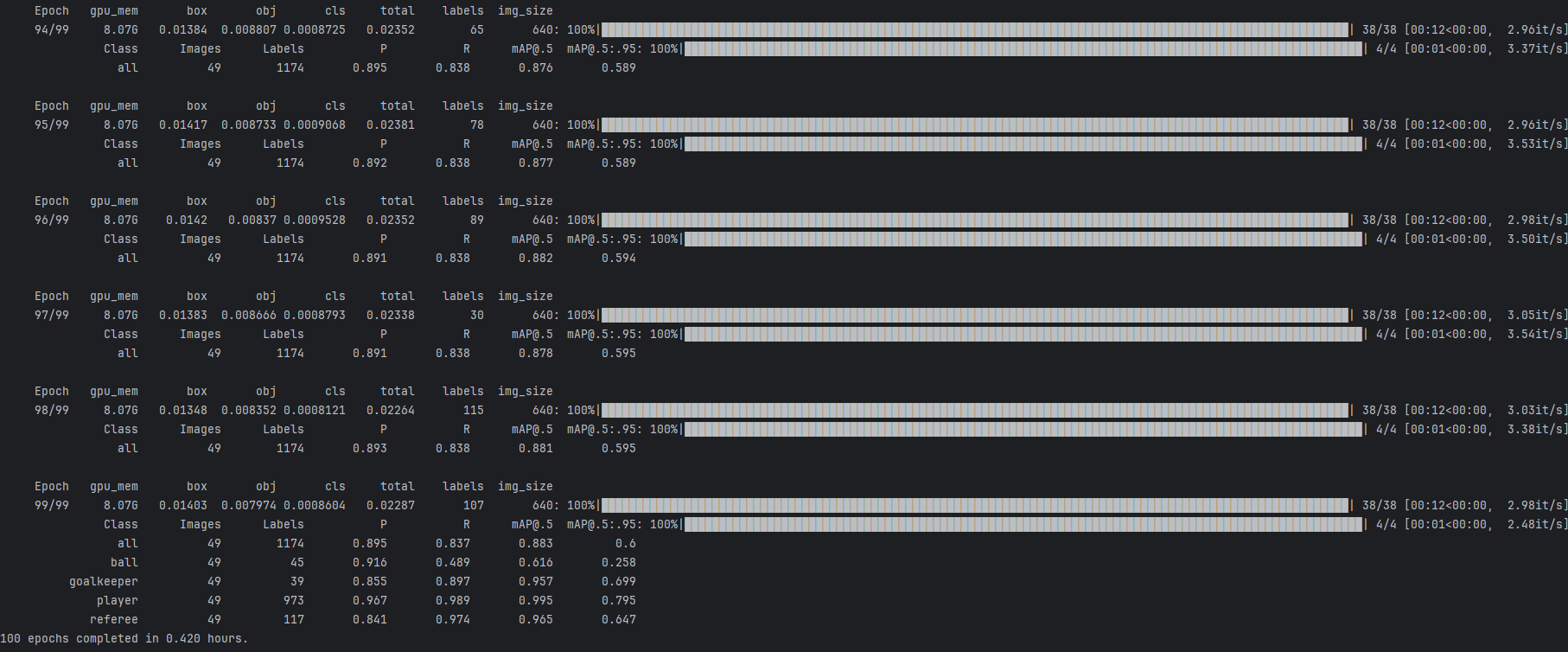
loss\_ota: 1 # use ComputeLossOTA, use 0 for faster training

1. Detección de Imágenes:

Para realizar la detección de categorías en una imagen, se utilizó el siguiente comando:

python detect.py --weights runs/train/football\_detection3/weights/best.pt --source football-players/detect/jugadores4.jpg





**Resultados Obtenidos:**

El modelo entrenado logró detectar exitosamente las categorías de 'ball', 'goalkeeper', 'player' y 'referee' en las imágenes de prueba. La precisión y el recall del modelo fueron satisfactorios, logrando una alta tasa de detección en distintas condiciones de iluminación y ángulos de cámara.

Resultado de las detecciones ejecutadas:

Un estadio de fútbol

Descripción generada automáticamente

Un juego de beisbol con espectadores en las gradas de un estadio

Descripción generada automáticamente

Un grupo de personas en una cancha de fútbol

Descripción generada automáticamente

**Conclusiones:**

1. El uso de la arquitectura YOLOv7 demostró ser efectivo para la detección de objetos en el contexto de un partido de fútbol. La reutilización de los pesos pre-entrenados en el dataset COCO permitió un entrenamiento eficiente y resultados precisos.
2. La clase “player” fue la que mejor predicción nos dio con valores picos de 98% (0.9).
3. La predicción para la clase “referee” ha estado sobre el 80% (0.8) lo que no está nada mal para considerarlo como un buen resultado.
4. La clase “ball” ha obtenido en uno de los casos un porcentaje de 60% (0.6) lo que nos obligaría a reentrenar el modelo ajustando alguno de los hiperparametros.
5. En la categoría o clase “goalkeeper” se presentó un caso en el que nos clasificó como “player”. En este caso el modelo nos dio una predicción errada por lo que buscaremos reentrenar el modelo ajustando alguno de los hiperparametros.

**Recomendaciones:**

1. **Mejorar la Diversidad del Dataset:** Incluir imágenes con mayor variabilidad en términos de condiciones de iluminación, clima y ángulos de cámara para robustecer el modelo.
2. **Ajuste de Hiper-parámetros:** Continuar ajustando los hiper-parámetros para mejorar aún más la precisión del modelo.
3. **Pruebas en Tiempo Real:** Implementar el modelo en un sistema de detección en tiempo real para evaluar su desempeño en situaciones de juego en vivo.
4. Tengamos en cuenta ajustar este hiperparametro **momentum/Adam** ya que es considerado el más conocido para optimizar en redes neuronales profundas según el artículo **Understanding Deep Learning Optimizers: Momentum, AdaGrad, RMSProp & Adam**[1].

**Código Fuente:**

<https://drive.google.com/drive/folders/19TE_waBXRYnaEwuZVQLQ7VUPYY5mWTvS?usp=drive_link>

**Referencia:**

1. https://towardsdatascience.com/understanding-deep-learning-optimizers-momentum-adagrad-rmsprop-adam-e311e377e9c2