

# Military Aircraft Recognition



**Alumnos:** Bautista Priano Sobenko, Jeremias Savarino

**Profesor:** Gustavo Meschino

**Materia:** Inteligencia Artificial

**Fecha de entrega:** 29/11/2022

# Red convolucional para detección de objetos con bounding boxes

## Introducción

### **¿Qué es la detección de objetos?**

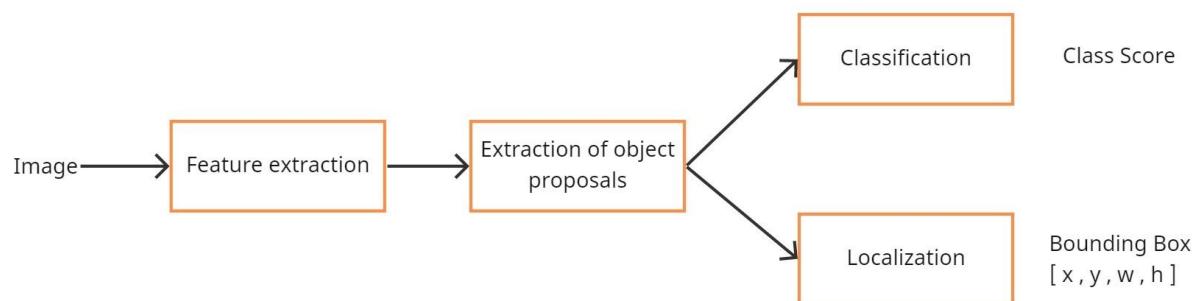
La clasificación de imágenes implica asignar una etiqueta de clase a una imagen, mientras que la localización de objetos implica dibujar un cuadro delimitador alrededor de uno o más objetos en una imagen. La *detección de objetos* es más desafiante y combina estas dos tareas y dibuja un cuadro delimitador (bounding box) alrededor de cada objeto de interés en la imagen o video y les asigna una etiqueta de clase, lo que permite identificar y ubicar dónde están dichos objetos (o como se mueven a través de) una escena determinada.

Los algoritmos diseñados para realizar la detección de objetos se basan en dos enfoques:

1. One-stage object detection
2. Two-stage object detection

### **Two-stage object detection**

Enfoque en donde la detección y el reconocimiento de objetos son dos procesos diferentes. Se utilizan dos modelos: un modelo se usa para identificar regiones que posiblemente contengan objetos, y un segundo modelo actuaría como clasificador utilizando estas regiones previamente identificadas. Estos métodos son relativamente lentos, pero generalmente más precisos que los One-stage.



*Métodos que utilizan este enfoque:*

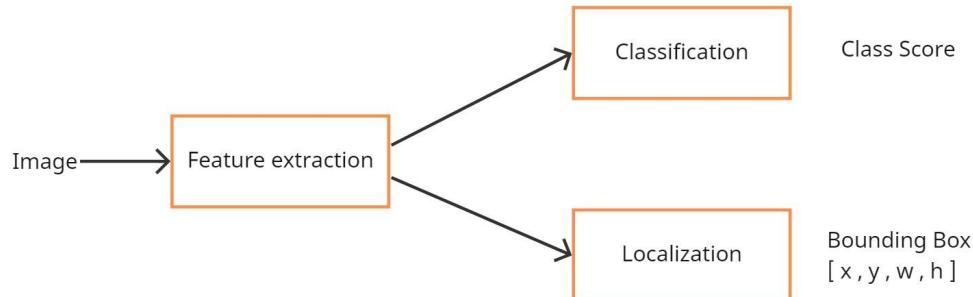
- RCNN : Region based CNN
- Fast RCNN : Faster version of RCNN
- Faster RCNN
- RFCN : Region based fully convolutional network
- Mask RCNN : Faster RCNN extension

### **One-stage object detection**

Enfoque en donde la detección y el reconocimiento de objetos ocurren al mismo tiempo.

Requiere solo un paso a través de la red neuronal y se obtendrá directamente la clasificación de objetos y la localización de la bounding box correspondiente.

La ventaja principal de este proceso es su gran rapidez y su alta velocidad de inferencia.



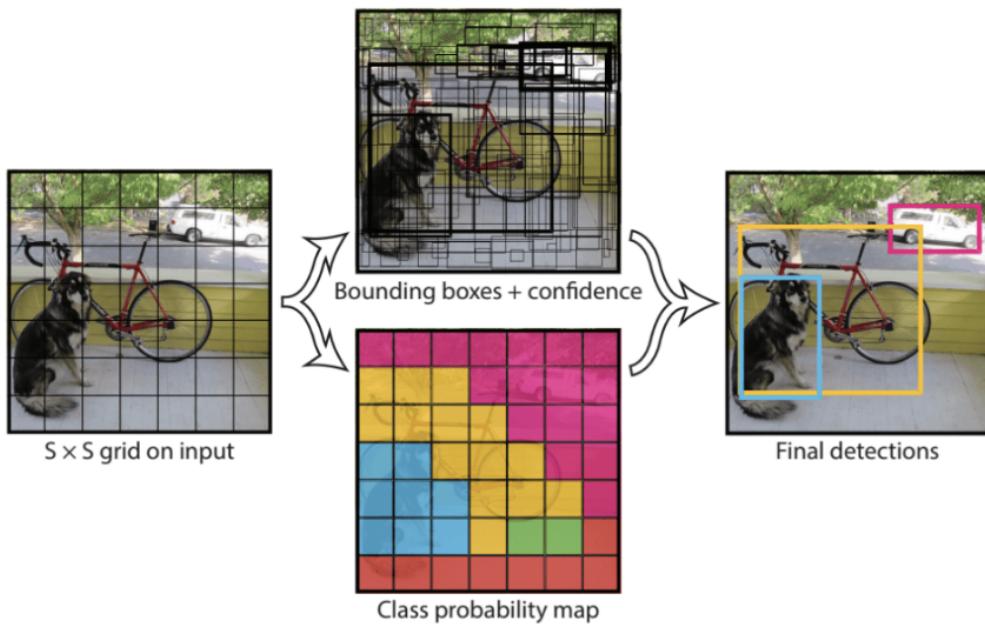
*Métodos que utilizan este enfoque:*

- **YOLO** : You only look once
- SDD : Single shot detector
- RetinaNet
- DCN : Deformable convolutional networks
- NAS-FPN : Feature Pyramid network

El método seleccionado para la resolución de este trabajo es el algoritmo con el enfoque One-stage, YOLO.

## YOLO: You Only Look Once

Este modelo fue el primero en tratar a la detección de objetos como un problema de regresión en donde la salida sea la probabilidad de la clase a la par de las coordenadas de la bounding box.



Funciona dividiendo primero la imagen de entrada en una cuadrícula de celdas, donde cada celda es responsable de predecir una bounding box si el centro de dicha bounding box cae dentro de la celda.

Cada bounding box incluirá las coordenadas x , y , el alto y el ancho y la confianza.

Se puede encontrar múltiples versiones desarrolladas a lo largo de los años desde el año 2015 hasta la actualidad.

Algunas de ellas son: YoloV1, YoloV2, YoloV3, YoloV4 , YoloV5, YoloV6, **YoloV7** , YoloX, YoloR, etc.

La versión con la que trabajaremos será la última, la versión 7, desarrollada e introducida a la familia YOLO este mismo año, en Julio del 2022.

## YOLOv7

Hoy en día está causando revolución en las comunidades de la inteligencia artificial, ya que este nuevo algoritmo supera a todos los modelos anteriores de detección de objetos y a las versiones anteriores de YOLO tanto en velocidad como en precisión y, además, requiere un hardware bastante más económico que otras redes neuronales y se puede entrenar mucho más rápido en conjuntos de datos pequeños sin pesos previamente entrenados. Estableció un punto de inflexión al llevar el rendimiento de la detección de objetos en tiempo real a un nivel superior.

## Arquitectura

YOLO no es una arquitectura única sino un modelo de investigación flexible escrito en lenguajes de bajo nivel. El modelo tiene tres componentes principales: la cabeza, el cuello y la columna vertebral. Diferentes conjuntos de componentes y arquitectura están asociados con los tres componentes mencionados anteriormente, lo que da lugar a diferentes versiones de YOLO.

- *The backbone (columna vertebral)* :
  - Es una red neuronal profunda compuesta principalmente por capas convolucionales. El principal objetivo de la “backbone” es la detección y extracción de características esenciales de la imagen. Generalmente este componente se encuentra pre-entrenado con un dataset específico como ImageNet o bien COCO.
- *Neck (cuello)*
  - Es una capa intermedia entre la Backbone y el Head. Toma las características extraídas (feature maps) por la columna vertebral y las procesa, fusionándose y mezclándolas para predecir las coordenadas de las bounding boxes y las probabilidades de clasificación.
- *Head (cabeza)*
  - El head es la capa de salida del modelo que se encargará de realizar las predicciones finales. Existirán múltiples heads y cada una tendrá funciones específicas. Por ejemplo, un head determinará la coordenadas de las bounding boxes, otro determinará cuando un objeto está presente o no dentro de un determinado frame, otro head detectara a qué clase corresponde el objeto.

## ¿Qué es lo nuevo de YOLO v7?

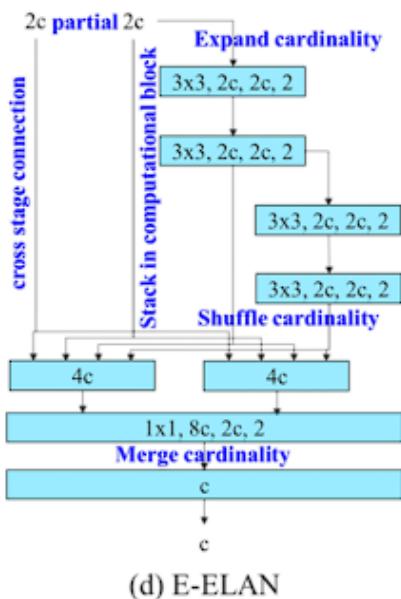
### **Improved backbone convolutional layer**

**(E-Elan : Extended efficient layer aggregation network)**

Conclusiones del paper: No cambia en absoluto la ruta de transmisión del gradiente de la arquitectura original, sino que utiliza la convolución de grupos para aumentar la cardinalidad de las características añadidas, y combina las características de diferentes grupos en forma de expandir, mezclar y fusionar cardinalidad. Esta forma de operar puede mejorar las características aprendidas por diferentes mapas de características y mejorar el uso de los parámetros y los cálculos.

Ha sido diseñada para mejorar la rapidez y la precisión de los siguientes factores:

1. Costo de acceso a la memoria
2. Relación de canales de E/S
3. Operación sabia del elemento
4. Activaciones
5. Trayectoria del gradiente



### **Trainable bag of freebies (BoF)**

Son distintos métodos que aumentan el rendimiento del modelo sin aumentar el costo de entrenamiento.

### **Planned re-parameterized convolution**

Es una técnica que se utiliza luego del entrenamiento para mejorar el modelo. Si bien aumenta el tiempo de entrenamiento, también mejora los resultados de inferencia. Fusiona múltiples módulos computacionales en uno en la etapa de inferencia. La técnica de re-parametrización del modelo puede considerarse como una técnica de conjunto, y podemos dividirla en dos categorías:

- A nivel de módulo: este tipo de método divide un módulo en varias ramas de módulos durante el entrenamiento , que pueden o no ser iguales, y las integra formando un módulo completamente equivalente durante la inferencia.
- A nivel de modelo: esta se puede realizar de dos maneras:
  - Usando diferentes datos de entrenamiento pero la misma configuración entrena varios modelos. Luego se promedian sus pesos para obtener el modelo final.
  - Tomando el promedio de los pesos de los modelos en diferentes épocas.

Sin embargo, hay que tener en cuenta que no todos los módulos re-parametrizados podrán ser perfectamente aplicados a las diferentes arquitecturas.

### **1. Coarse for auxiliary and fine for lead loss**

La supervisión profunda es una técnica que se usa a menudo en el entrenamiento de redes profundas. Su concepto principal es agregar cabezas auxiliares adicionales en las capas intermedias de la red, utilizando los pesos de estas últimas como guía.

Incluso para arquitecturas que convergen bien como ResNet y DenseNet, la supervisión profunda puede mejorar significativamente el rendimiento del modelo en muchas tareas.

En la arquitectura YOLOv7, la cabeza responsable del resultado final se denomina cabeza principal, y la cabeza que se utiliza para ayudar en el entrenamiento se denomina cabeza auxiliar.

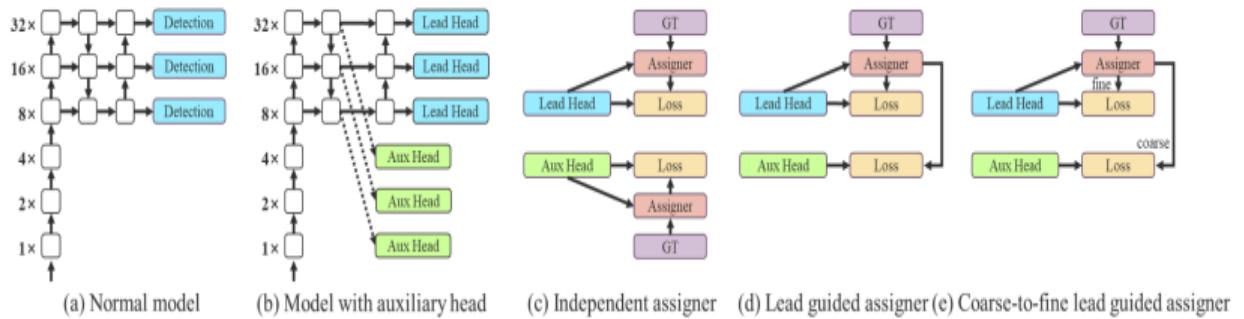
El label assigner es un mecanismo que considera los resultados de la predicción de la red junto con los targets y luego asigna etiquetas flexibles. Cabe destacar que el Label Assigner genera etiquetas finas y gruesas, en lugar de generar etiquetas duras.

YOLO v7 utiliza la predicción de la cabeza principal como guía para generar etiquetas ordenadas jerárquicamente, de gruesas a finas, que serán utilizadas tanto para el aprendizaje de las cabezas auxiliares, como para el aprendizaje de la propia cabeza principal.

Existen dos tipos de estrategias de supervisión profunda que utiliza este modelo:

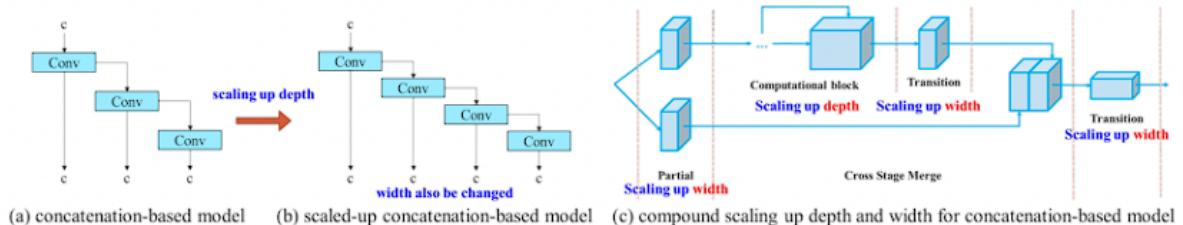
Asignación de la etiqueta guiada por la cabeza principal: al permitir que la cabeza auxiliar menos profunda aprenda directamente la información que la cabeza principal ha aprendido, la cabeza principal podrá concentrarse más en aprender la información residual que aún no se ha aprendido.

Asignación de etiqueta guiada por la cabeza principal de grueso a fino: este mecanismo permite ajustar dinámicamente la importancia de la etiqueta fina y la etiqueta gruesa durante el proceso de aprendizaje, y hace que el límite superior optimizable de la etiqueta fina sea consistentemente más alto que el de la etiqueta gruesa.



## Model scaling for concatenation-based models

El objetivo principal del escalado del modelo es ajustar algunos atributos y generar modelos de diferentes escalas para satisfacer necesidades de diferentes velocidades de inferencia. En YOLOv7 se escala la profundidad y ancho de la red al mismo tiempo que se concatenan capas.



## Entrenamiento del modelo

### Introducción

En este trabajo decidimos entrenar un modelo YOLO v7 que sea capaz de detectar, localizar e identificar distintos aviones militares. Es un modelo de clasificación donde existen 39 clases diferentes.

Como entrada este modelo recibe una imagen o video y como salida devuelve la misma imagen o video, pero con el bounding box correspondiente encuadrando al avión e indicando tanto el nombre de la clase como la probabilidad de pertenencia a la misma.

## **Dataset**

El dataset utilizado fue obtenido de la página de Roboflow, una página web que permite explorar y trabajar con más de 100.000 datasets diferentes. Hemos seleccionado el dataset de nombre “military-aircraft-detection” constituido por 5152 imágenes. En este dataset se utiliza la técnica de “Data augmentation” agregando imágenes que varían en brillo, color, orientación, etc.

El dataset se encontrará dividido en 3 carpetas principales: Train (70%) , Validation (20%), Test (10%).

En cada carpeta se encuentran distintas imágenes que el modelo utilizará para entrenar, validar y testear, respectivamente.

Cada imagen perteneciente a la carpeta “Train” y “Validation” tiene su etiqueta YOLO, asociada a la bounding box correspondiente. La etiqueta se encontrará compuesta por: Confianza, Coordenadas X e Y , Altura H y Ancho W : [ p, x, y, h, w ]

Se deben tener idealmente,a la hora de entrenar el modelo, la misma cantidad de imágenes aproximadamente de cada clase considerada. En este dataset eso no se cumple, por lo tanto, si se quisiera mejorar el modelo, un aspecto a considerar sería quitar algunas imágenes excedentes de algunas clases, y por otro lado agregar más imágenes de otras clases que lo requieran, con sus respectivas etiquetas, con el objetivo de equilibrar el dataset.

## **Entrenamiento**

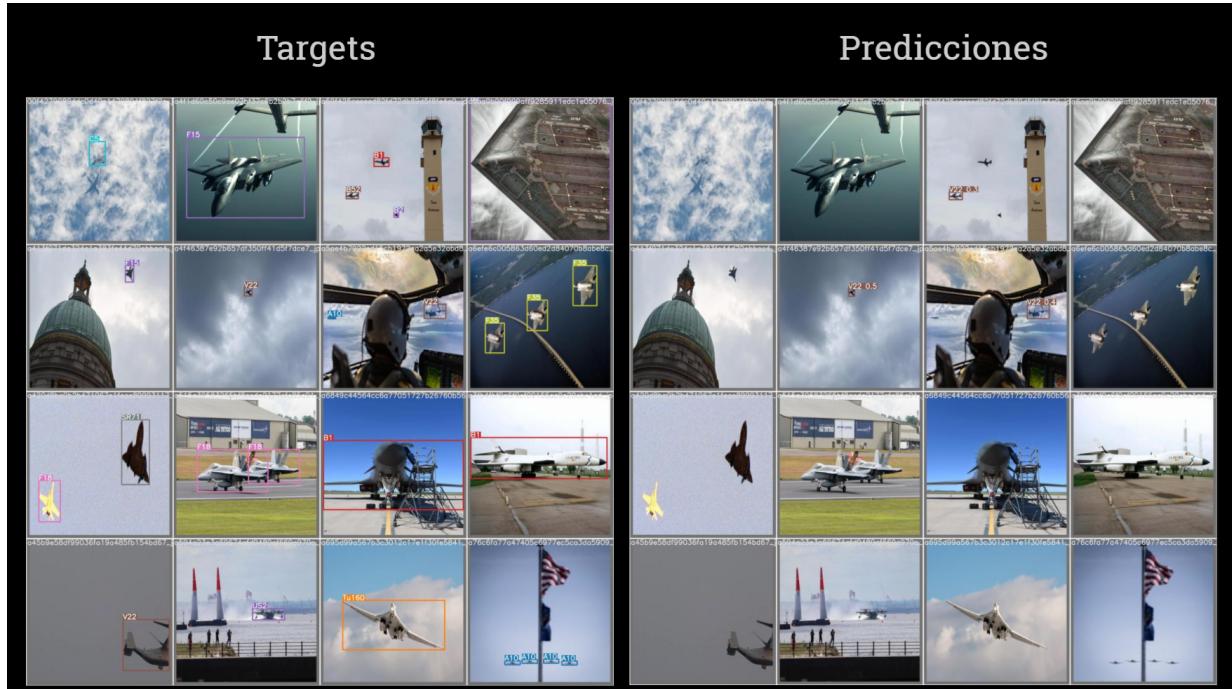
A la hora de realizar el entrenamiento de la red, fuimos tomando etapas de a 10 y de a 20 épocas,con batches de 16 imágenes, y en base a los resultados y los gráficos obtenidos, se fueron ajustando los distintos hiper parámetros del modelo para encontrar mejoras,tanto en la precisión como en el tiempo de entrenamiento de cada época.

Cabe destacar que se probó utilizar batches de 32 imágenes y de 8 imágenes, pero no dieron resultado.

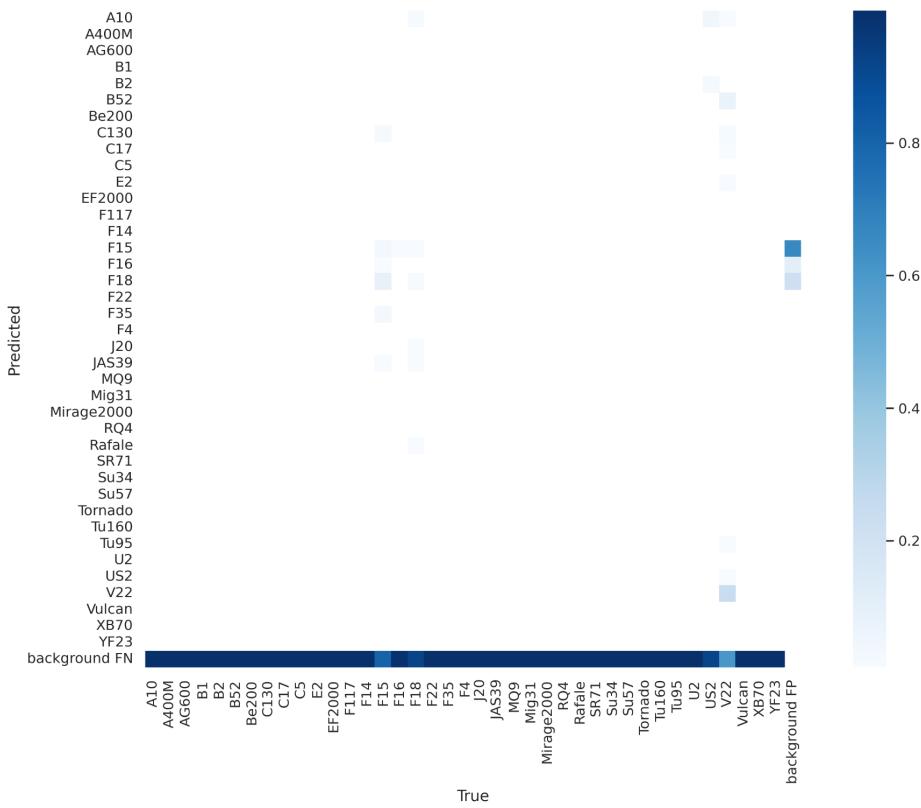
### **Primer entrenamiento: 10 epochs**

En estas primeras épocas se obtuvieron resultados poco precisos, esto se puede ver reflejado tanto en el valor de la precisión ( 0.21 en la mejor época) como en el recall obtenidos luego de este primer entrenamiento. El tiempo de entrenamiento resultó bastante elevado en esta primera etapa, en donde cada una de las épocas necesitaba de 10/11 minutos para ser entrenada.

El modelo logra localizar muy pocos aviones en las imágenes.



Matriz de confusión luego del primer entrenamiento



La pésima precisión mencionada se puede apreciar en esta matriz, donde apenas se puede ver la diagonal marcada en algunas clases, con una precisión muy pobre, y se pueden distinguir varias identificaciones falsas. Es un resultado esperado ya que fue el primer entrenamiento, realizado con los parámetros por defecto, y se entrenaron pocas épocas.

## Cambios

En base a estos deficientes resultados decidimos freezar algunas capas con el objetivo de evitar que los pesos cambien dentro de las capas elegidas (50 capas convolucionales pertenecientes al backbone del modelo) y así lograr reducir el tiempo de entrenamiento de cada época.

Si bien se pudo notar una leve mejora tanto en el tiempo de entrenamiento de cada época (cayó de 11 min a 10 min) como en los valores de la accuracy (0.21 en la mejor época), los resultados seguían siendo lejanos a los deseados.

Debido a esto decidimos modificar el learning rate para encontrar una mejora en el próximo entrenamiento. El LR pasó de ser 0.1 a ser 0.2.

Además detectamos un error clave en la configuración del entorno de trabajo, cuya corrección permitió mejorar el tiempo de entrenamiento: modificamos el poder suministrado por la GPU para que esta nos otorgue su máximo rendimiento.

En el tercer entrenamiento se pudo apreciar una mejora respecto al entrenamiento anterior. La precisión aumentó de 0.2 a 0.29 y en la matriz de confusión se empezó a notar la

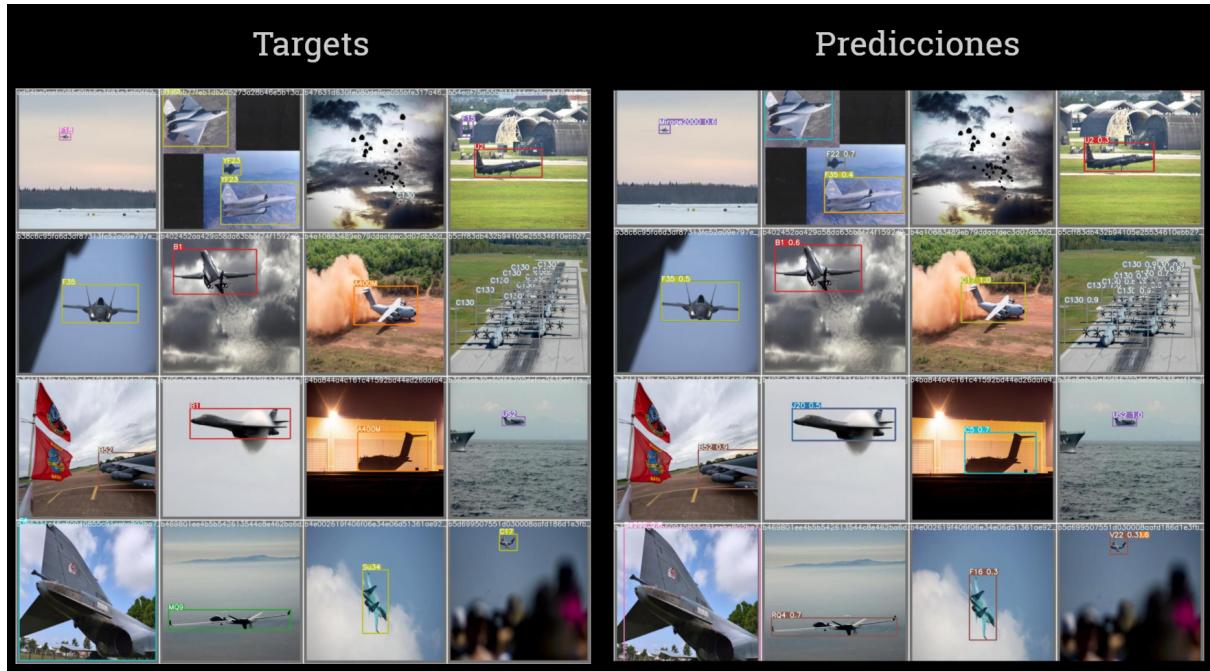
diagonal principal, lo cual como sabemos es una gran señal. Por ello se procedió a continuar con el entrenamiento, sin necesidad de modificar los hiper parámetros.

La cantidad de épocas entrenadas, para este momento era de 80.

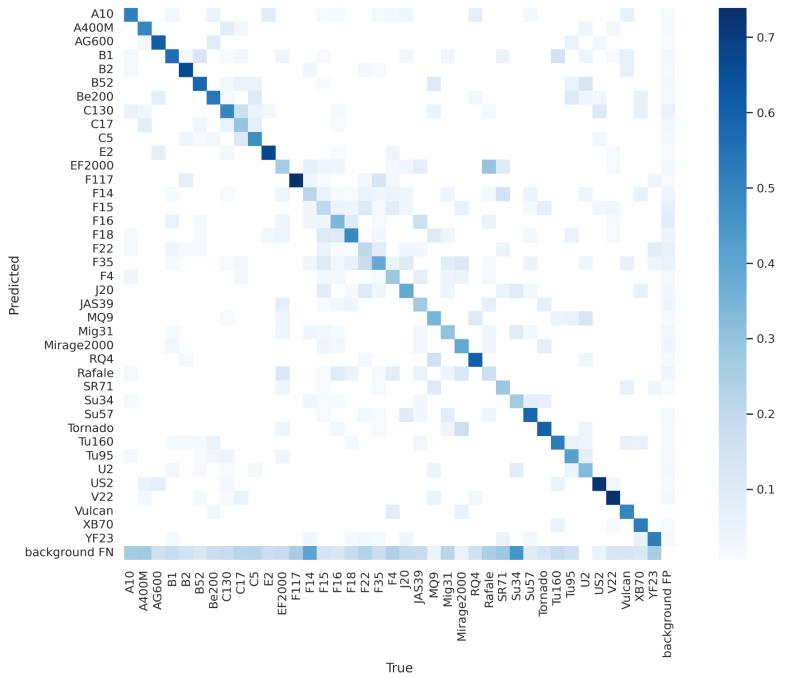
#### Cuarto entrenamiento: 30 epochs

Luego de esta etapa, se pudo notar un aumento más que notable en el valor de la precisión, con casi un 100% de incremento, pasando de 0.29 (en el tercer entrenamiento) a 0.58. El tiempo de entrenamiento cayó a 9 minutos por época.

Si bien el modelo a este punto logra localizar a los aviones en las imágenes, utilizando las correspondientes Bounding Boxes, falla a la hora de clasificarlos.



**Matriz de confusión luego del cuarto entrenamiento**

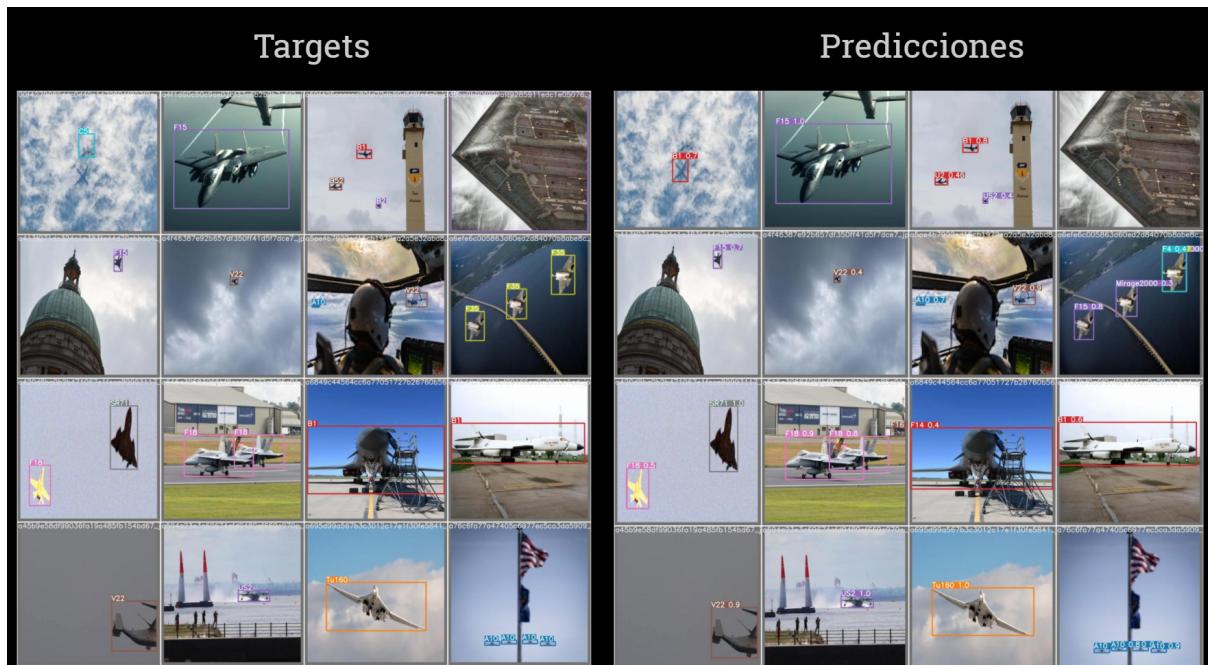


En la matriz de confusión se observa una gran coloración sobre la diagonal principal, dando indicios de que el modelo se encuentra prediciendo de forma adecuada, en donde los resultados obtenidos son los esperados.

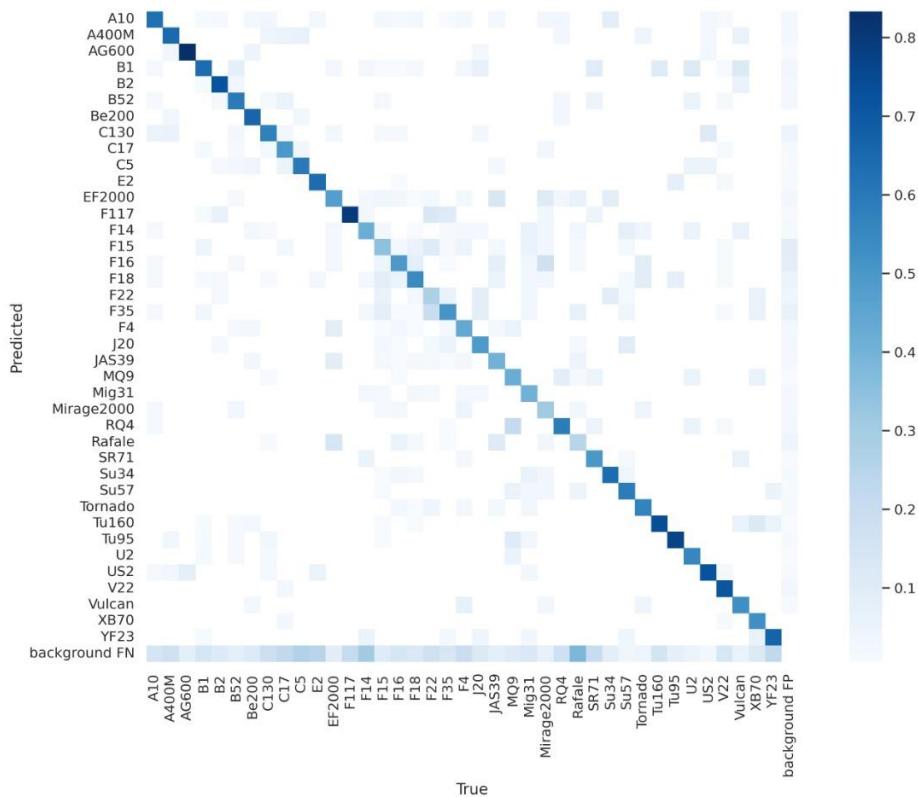
Cabe aclarar que todavía se observan bastantes falsos positivos dentro de la matriz.

### Último entrenamiento: 20 epochs

En este último entrenamiento, se logró disminuir el tiempo de entrenamiento a 7.5 min por cada época entrenada. Se logró una precisión de 0.68. A este punto el modelo es capaz tanto de localizar como de identificar correctamente la mayoría de los aviones.



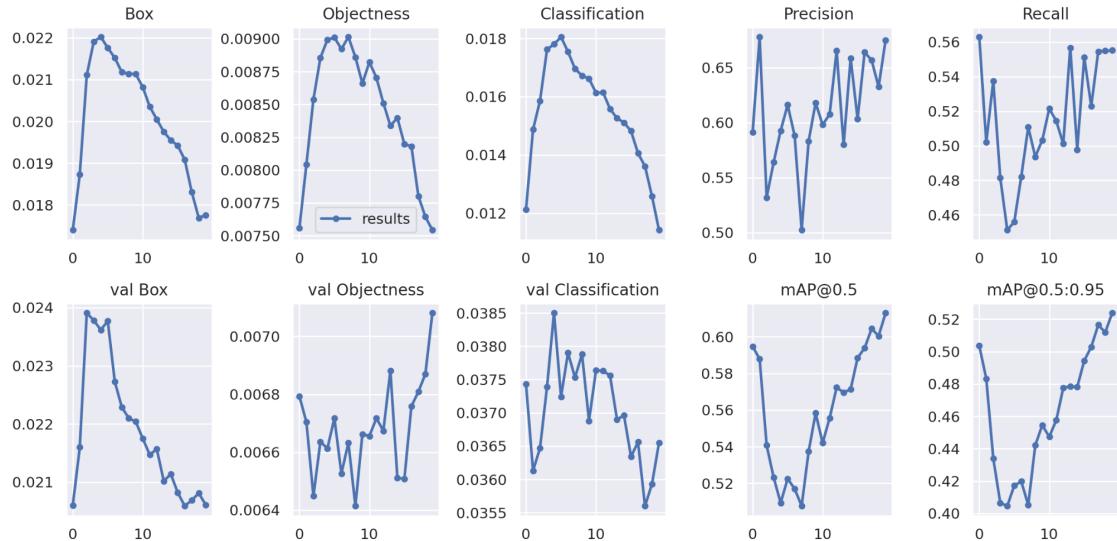
## Matriz de confusión luego del último entrenamiento



En la matriz de confusión se puede observar como la diagonal está marcada con azul oscuro, y como se redujeron notablemente la cantidad de falsos positivos respecto a los anteriores entrenamientos.

## Métricas

A continuación se pueden observar diferentes métricas generadas por el modelo, las cuales nos ayudarán a observar las mejoras a lo largo de los entrenamientos. Se explicarán, en forma breve, cada una de ellas.



- **Box loss:** Representa que tan bien el algoritmo puede localizar el centro de un objeto
- **Objectness Loss:** Representa el error de IoU (Intersection Over Union)
- **Classification loss:** Representa que tan bien el algoritmo puede predecir la clase correcta de un determinado objeto. “Cross Entropy Loss”
- **Precision:** Es la habilidad que tiene un modelo para identificar solo los objetos relevantes. Permite conocer la calidad del modelo en la clasificación. Responde a la pregunta: ¿Qué proporción de las detecciones es realmente correcta?

$$Precision = \frac{\text{Correct positive guesses}}{\text{Total positive guesses}} = \frac{TP}{TP + FP}$$

- **Recall:** Nos informa sobre la cantidad de casos correctos que el modelo es capaz de identificar. Permite conocer de las clases que tiene el modelo cuales fueron identificadas correctamente.

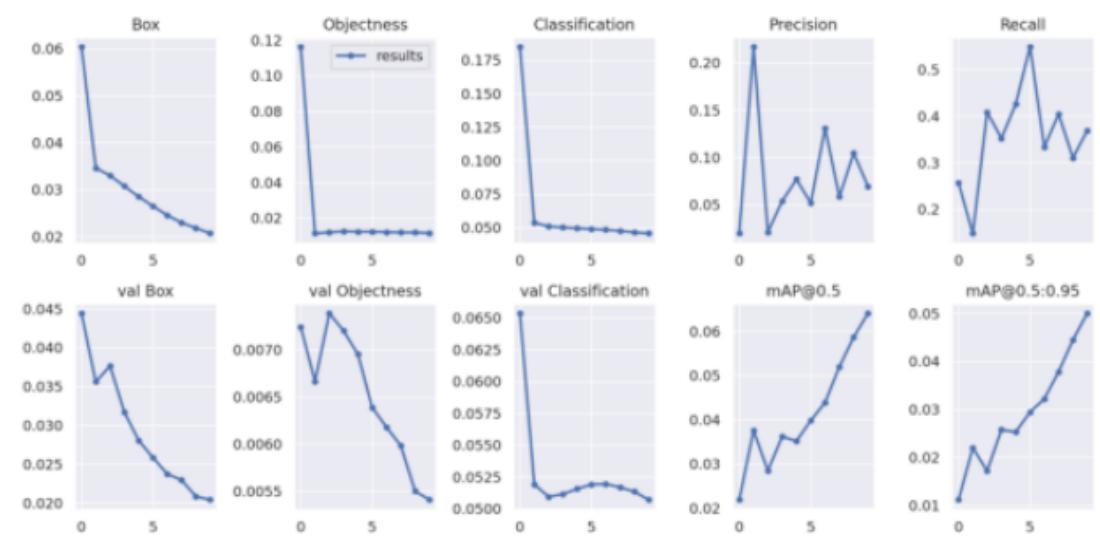
$$Recall = \frac{\text{Correct positive guesses}}{\text{All positive labels}} = \frac{TP}{TP + FN}$$

- **mAP:** representa el promedio de los AP calculados para todas las clases, siendo AP área que se encuentra bajo la curva de precisión-recall . Un mAP óptimo indica que el modelo es estable y consistente a través de los diferentes umbrales de confianza  
 $mAP@0.5$  (Umbral de confianza del IOU de 0.5)  
 $mAP@0.5:0.95$  (Umbral de confianza del IOU dentro del rango [0.5 , 0.95] )

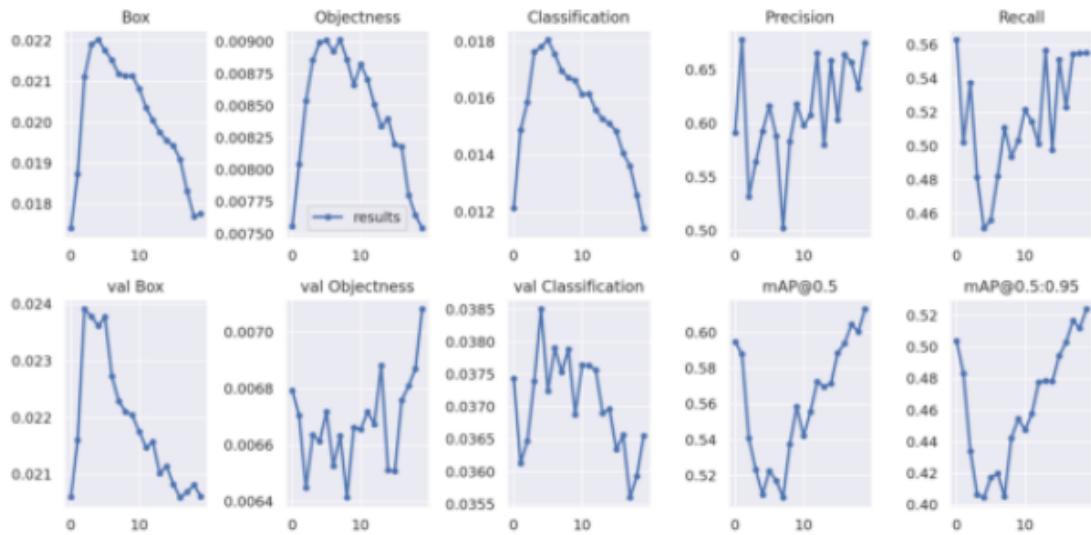
- **F1:** Es la media armónica entre precisión y recall. A mayor valor de precisión y recall, mayor resultará el valor F1.

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

## Primer entrenamiento



## Último entrenamiento



Viendo las distintas métricas, tanto del primer como del último entrenamiento, se puede observar una gran mejora del sistema.

Las métricas de error( Box, Objectness, Classification) se redujeron notablemente con el pasar de los entrenamientos, mientras que las métricas de precisión, recall y mAP aumentaron considerablemente.

En ellas se ve reflejada la mejora del sistema a lo largo de las épocas.

## Conclusiones

Si bien la precisión es cercana al 70% se encontraron diferentes aspectos no considerados que pueden llegar a mejorar en forma significativa el modelo, tanto en la rapidez de entrenamiento, como en la precisión.

Uno de estos aspectos es el *balance del dataset*, en donde debemos igualar el número de imágenes por clase, eliminando así clases con sobre-muestreo y clases con sub-muestreo. El sobre-muestreo puede causar overfitting y enlentecer el entrenamiento innecesariamente; por otra parte en el sub-muestreo podemos perder información valiosa. Este cambio impactaría en forma significativa sobre la precisión y el recall.

Otro de los aspectos importantes a considerar es el *re-escalado de imágenes*, utilizado para disminuir tiempos de entrenamiento. Con esto se conseguiría una caída importante en los tiempos de entrenamiento, pero corriendo el riesgo de perder calidad en el reconocimiento del objeto, y por ende perder precisión.



## Bibliografía: 📚

- **Paper de YOLOv7 : “Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”:** <https://arxiv.org/pdf/2207.02696.pdf>
- <https://viso.ai/deep-learning/yolov7-guide/>
- <https://vishal-ai.medium.com/yolov7-making-yolo-great-again-7b1ec1f6a2a0>
- <https://learnopencv.com/yolov7-object-detection-paper-explanation-and-inference/>
- <https://blog.roboflow.com/yolov7-breakdown/>
- **Enlace al Collab:**  
<https://drive.google.com/file/d/1UMhVqMVoKKzMsaHpj-vEOEQP15S2kS9B/view?usp=sharing>