

# **PRIMERA ENTREGA DE PROYECTO IA**

Presentado por:

Carlos Eduardo Báez Coronado

Laura Catalina González Velandia

Profesor:

Raúl Ramos Pollan



Universidad de Antioquia

Facultad de Ingeniería

Departamento de Ingeniería Mecánica

Medellín, 5 de julio del 2022

## 1. Descripción del problema

Nuestro propósito en este proyecto es implementar una red neuronal que pueda clasificar distintos objetos en una imagen, y también predecir la localización de las Bounding Boxes de una cantidad limitada o especificada de objetos que se puedan encontrar en la carretera, tales como automóviles, motocicletas, bicicletas, trenes, etc. Tomando como punto de partida el dataset de Open Images V6 <https://storage.googleapis.com/openimages/web/index.html>.

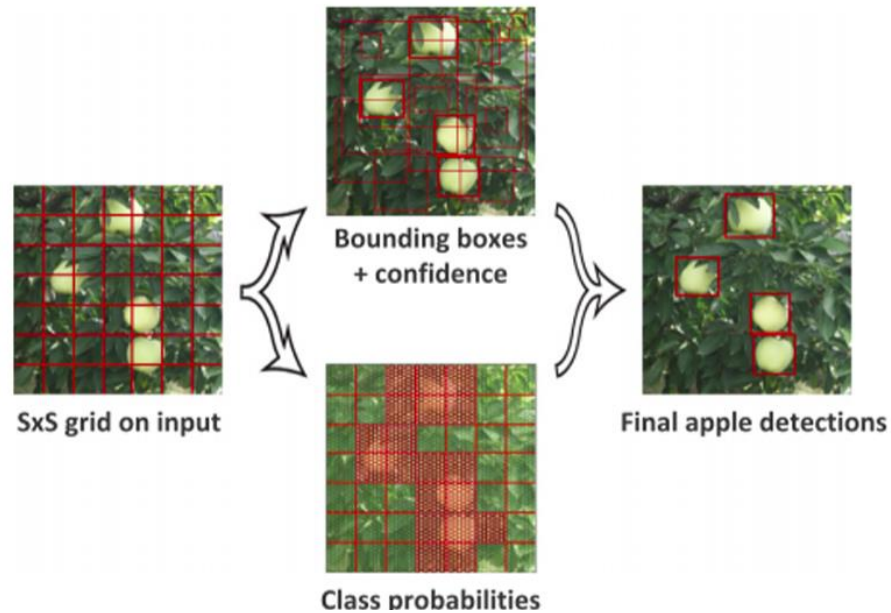
Este Dataset está formado por aproximadamente 16 millones de bounding boxes para más de 600 categorías; por tanto, dado que no se poseen los recursos computacionales suficientes para realizar este trabajo completamente, solo se trabajará con una pequeña porción de los datos, aplicando transfer learning a una red neuronal para aprovechar arquitecturas ya entrenadas y muy bien definidas.

## 2. Dataset

Para el entrenamiento de una red neuronal es necesario tener en cuenta un dataset que contenga imágenes que permitan una debida detección de objetos, donde entre mayor sea la cantidad de imágenes, mayor será la eficacia del entrenamiento.

Dado que el objetivo principal de este proyecto es conectar la red neuronal a una cámara para tener en tiempo real los datos recogidos en una vía, se debe de seleccionar un método que se desempeñe de manera adecuada en *entornos outdoor*, donde encontramos el algoritmo YOLO “You Only Look Once”, método el cual recibe como entrada una imagen y devuelve como salida los bounding boxes de los objetos que ha clasificado, tratando el problema de detección de imágenes como un problema de regresión; además al correr una sola red neuronal convolucional para realizar todas sus tareas, donde esto trae beneficios tanto en velocidad (proceso de convolución se realiza una sola vez en toda la imagen para realizar las predicciones), por lo que se pueden procesar en tiempo real un video con unas métricas de predicción y clasificación.

Además, hay reducción de errores de fondo, ya que se mira la imagen de manera entera, al dividirla en una malla  $S \times S$ , y no por secciones; donde cada celda predice un número  $B$  de bounding boxes y lo que se conoce como confianza, que representa la probabilidad de que ese objeto sea el que se ha predicho, por lo que se pueden sacar conclusiones globales, lo que disminuye los errores de falsas predicciones.

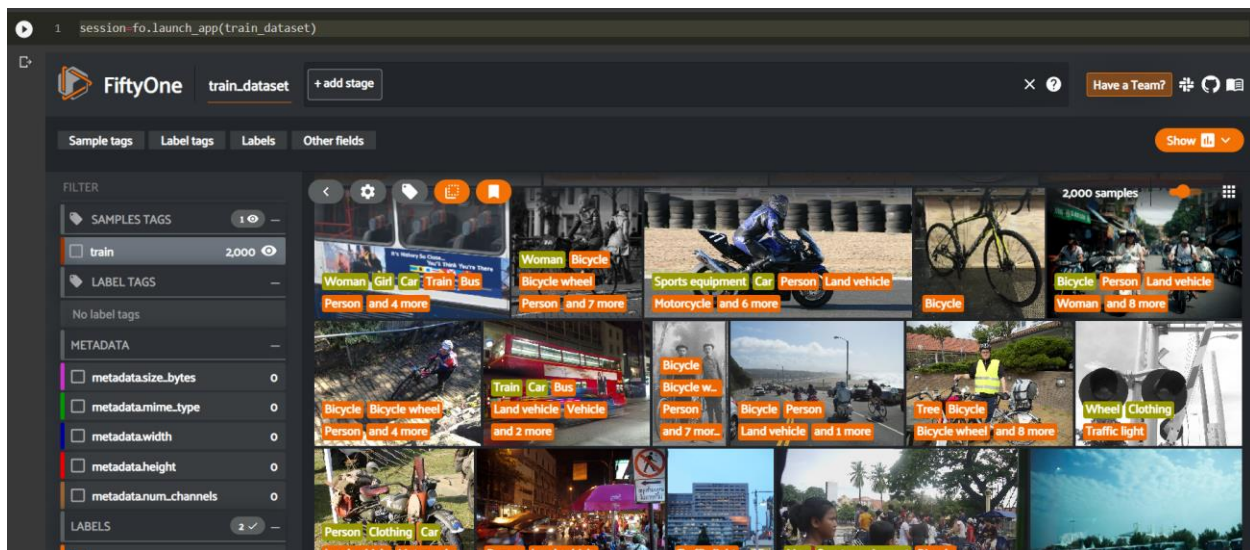


*Ilustración 1: Descripción del funcionamiento de YOLO.*

Las imágenes las vamos a conseguir haciendo uso de la librería FiftyOne, con la cual podemos hacer consultas y filtros avanzados en el dataset de Open Images de Google. Específicamente utilizaremos el FiftyOne Dataset Zoo, que contiene una colección de los datasets más comunes que los podemos cargar con comandos simples. Para hacer las etapas de entrenamiento, validación y testeo, vamos a crear varios subsets que contengan los datos para cada etapa. Por ejemplo, para la importación del dataset de entrenamiento lo vamos a hacer de la siguiente manera:

```
[5] 1 train_dataset = foz.load_zoo_dataset(
2     "open-images-v6", #Dataset a partir del cual vamos a tomar los datos
3     split=('train'), #Este subset solo va a estar compuesto por los datos de entrenamiento
4     label_types=["detections", "classifications"], #La red neuronal va a estar compuesta por localización y clasificación
5     classes = ["Can", "Bus", "Ambulance", "Traffic light", "Bicycle", "Stop sign", "Motorcycle"],
6     max_samples=2000, #Imágenes que va a tener el dataset de entrenamiento
7     shuffle=True,
8     dataset_name='train_dataset'
9 )
```

Una de las ventajas que nos permite tener la librería de FiftyOne es la visualización de los datos de nuestro dataset. Para iniciar la aplicación y ver los datos cargados ejecutamos la línea que se muestra en la imagen de abajo:



Tal como podemos ver, cada imagen tiene dentro de sí varios objetos pertenecientes a distintas clases, por lo tanto, el dataset está cumpliendo los requerimientos de variables categóricas. Sin embargo, más adelante es posible que se reduzcan algunas características del dataset si el consumo computacional es muy alto.

### 3. Métricas de desempeño


Durante el entrenamiento de la red neuronal, vamos a utilizar varias métricas de desempeño según la tarea que deseemos realizar. La red que vamos a entrenar se va a encargar de la detección de objetos implementando un transfer learning de un modelo preentrenado de la arquitectura de YOLO (You Only Look Once) y también de la predicción de las bounding boxes; dicho modelo se puede encontrar en la página oficial de TensorFlow, o también en repositorios de GitHub o competencias de Kaggle.

En esta entrega solo vamos a profundizar sobre la métrica de desempeño empleada para la predicción de las cajas, pasando de largo sobre las otras métricas utilizadas durante el Forward y Backward Propagation para optimizar la clasificación de las imágenes.

#### - **Intersection Over Union:**

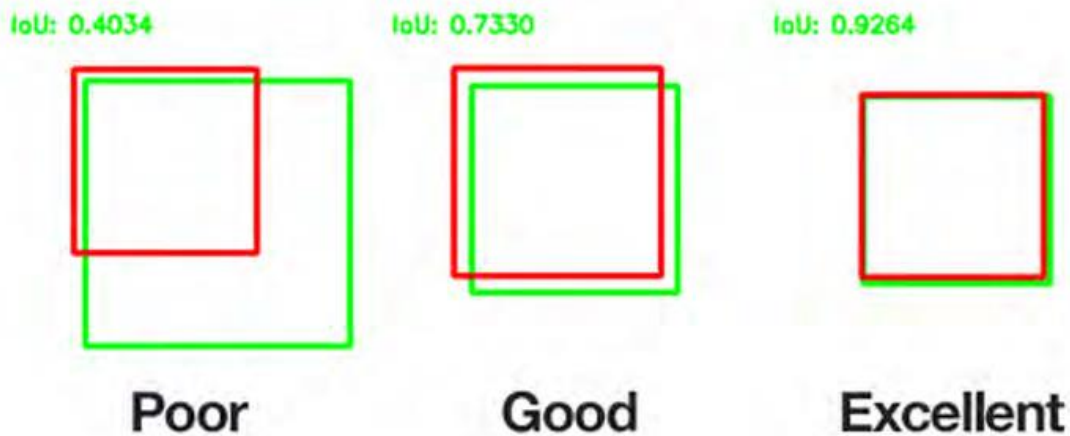
Esta métrica es una de las más utilizadas para medir la precisión de un detector de objetos frente a un dataset preestablecido, y se sitúa por encima de MSE como función de costo durante el entrenamiento del modelo. Cuando se aplica IoU se está midiendo el área de superposición entre la localización de la caja predicha y la caja objetivo, dividido por el área de Unión de las dos cajas, es decir, esta toma el conjunto A de píxeles propuestos del objeto y el conjunto de píxeles verdaderos del objeto B y calcula:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Con esta métrica, durante el Forward propagation se calculará el costo de las predicciones, y durante el backward propagation se optimizarán los pesos e interceptos de cada neurona para mejorar la eficacia de la red neuronal.

El algoritmo premia las predicciones que mejor superpongan las verdades absolutas, donde únicamente se consideraran predicciones exitosas, cuando esta sea mayor o igual al 50%. En la siguiente Ilustración se muestran diferentes overlapping y los puntajes que IoU les asigna.



#### - mAP: mean Average Precision

Al tener una alta capacidad de generalización, el algoritmo es capaz de predecir con rapidez, pero se verá afectada la precisión de localización de los objetos; esto principalmente debido a que, si en una misma celda del mallado se encuentra más de un objeto, este no va a ser capaz de detectarlos todos. Principalmente al iterar una imagen y aplicar la CNN pre-entrenada para ver si detecta algo, nos encontraremos con las siguientes dificultades:



- Al movernos pocos píxeles con la ventana, el algoritmo estará detectando el mismo objeto múltiples veces, por lo que el tiempo de cómputo podría ser muy largo, pues para cada movimiento implica realizar una clasificación individual con la CNN.
- Lograr distinguir entre distintos objetos si estos se encuentran muy cercanos.

Debido a esto, surge la necesidad de crear una nueva métrica específica para la detección de imágenes en donde podamos evaluar al mismo tiempo si la clase de objeto es correcta y si la posición del “bounding box” (X, Y, alto y ancho) es buena, donde la precisión media para la clase C se puede calcular de la siguiente manera:

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)}$$

Donde:

- True Positive TP (c): se hizo una propuesta para la clase C y en realidad había un objeto de la clase C.
- Falso positivo FP (c): se hizo una propuesta para la clase C, pero no hay ningún objeto de la clase C.

#### 4. Desempeño

Esperamos que la red neuronal logre realizar la detección y clasificación de los objetos propuestos en tiempo real, con una precisión alta, y que además logre diferenciar y localizar distintos objetos al mismo tiempo dentro de una sola imagen, y de esta manera poder almacenar los datos para un estudio posterior de ellos. Dado que se van a adaptar modelos preentrenados, esperamos que la precisión en clasificación sea mayor al 90% y en la predicción de las cajas, sea superior al 50%.

## Bibliografía

Amorós, J. C. (julio de 2020). Detección basada en datos 3D de objetos urbanos. Alicante, España.

*Aprende Machine Learning*. (21 de agosto de 2020). Obtenido de <https://www.aprendemachinelearning.com/modelos-de-deteccion-de-objetos/>

Camilo Andrés Mosquera Victoria, G. A. (2020). Detección y seguimiento de múltiples objetos en tiempo real para vehículos autónomos. Cali, Colombia.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly.

<https://arxiv.org/abs/1506.02640><https://arxiv.org/abs/1506.02640>