INGENIERÍA EN SISTEMAS

FACULTAD DE INGENIERÍA

UNAH
UNIVERSIDAD NACIONAL AUTÓNOMA DE HONDURAS

DEPARTAMENTO DE INGENIERÍA EN SISTEMAS

INFORME PROYECTO

ING. HECTOR EMILIO GUEVARA PINTO

INTELIGENCIA ARTIFICIAL

IS701 - 1000

ESTUDIANTES

RUBEN NABIL DIAZ ROSALES	**-**	**20182400156**
ESTIVEN JOSE MEJIA RODRIGUEZ	**-**	**20201002454**
BRYAN STEVEN MEJIA CONTRERAS	**-**	**20221001381**

CIUDAD UNIVERSITARIA

TEGUCIGALPA, FRANCISCO MORAZÁN 24/03/2024

Índice

Contenido

Índice de ilustraciones

Introducción

La detección de objetos es una tarea fundamental en el campo de la visión por computadora, y tiene aplicaciones en áreas como seguridad, vehículos autónomos, reconocimiento de imágenes médicas y realidad aumentada, entre otras. A medida que la tecnología avanza, los modelos de redes neuronales (RNA) han mejorado significativamente en su capacidad para detectar y clasificar objetos en imágenes y videos en tiempo real.

En este contexto, YOLO (You Only Look Once) se ha consolidado como una de las arquitecturas más populares para la detección de objetos, debido a su alta velocidad y precisión. Existen varias versiones de esta arquitectura, entre ellas YOLOv5 y YOLO11, que siguen evolucionando para mejorar el rendimiento en diversas tareas de detección.

Este proyecto tiene como objetivo generar un modelo para detectar objetos en imágenes en tiempo real y detectar objetos en fotos así mismo comparar dos versiones de la arquitectura YOLO, específicamente YOLOv5 y YOLO11, en términos de su capacidad de detección de objetos. A través del entrenamiento de ambos modelos y la comparación de sus resultados, se busca determinar cuál de ellos es más eficiente y preciso en la tarea de detectar objetos en un conjunto de datos de imágenes.

Objetivos

Objetivo General

- Crear un modelo que sea capaz de detectar objetos en imágenes en tiempo real y objetos en fotografías.

Objetivos Específicos

- Comparar el rendimiento de los modelos YOLOv5 y YOLO11 en términos de precisión y recuperación al realizar tareas de detección de objetos.

- Evaluar la efectividad de cada modelo usando un conjunto de datos de imágenes con 8000 fotos de entrenamiento y 3000 fotos de validación para YOLOv5, y 10,000 fotos de entrenamiento y 5000 fotos de validación para YOLO11.

- Realizar un análisis comparativo de las métricas de rendimiento entre ambos modelos, evaluando su capacidad para detectar objetos correctamente en diferentes tipos de imágenes.

- Discutir las limitaciones y las oportunidades de mejora, especialmente debido a que ambos modelos fueron entrenados sin GPU.

Marco Teórico

El Problema a Resolver

El problema que se busca resolver en este informe es la detección de objetos en imágenes. Dada la creciente demanda de sistemas automatizados para identificar y clasificar objetos de manera rápida y precisa, es fundamental desarrollar modelos que puedan reconocer distintos objetos en imágenes o videos en tiempo real. Este problema tiene múltiples aplicaciones, desde la seguridad pública, donde se necesitan sistemas capaces de identificar personas o vehículos en imágenes de cámaras de vigilancia, hasta vehículos autónomos, que deben detectar y clasificar otros vehículos, peatones, señales de tráfico, etc., en su entorno para tomar decisiones en tiempo real.

La Arquitectura que se Utilizó

YOLO es un modelo de red neuronal convolucional (CNN) que tiene una arquitectura única y eficiente para la detección de objetos en imágenes. YOLO divide el problema de detección de objetos en un único problema de regresión, donde la red predice las coordenadas de los bounding boxes y las clases de los objetos, todo en una sola pasada.

Entrada y Salida

Entrada

- El modelo YOLO tiene una sola entrada, que es la imagen que se va a procesar. Esta imagen se pasa a través de una serie de capas convolucionales, que extraen características de la imagen.
- La entrada es una imagen de tamaño fijo, usualmente de dimensiones 416x416 o 640x640 píxeles, que se ajusta dependiendo de los parámetros de entrenamiento.

Salida

- La salida es un vector que contiene las predicciones de la red para cada celda de la imagen. Este vector incluye:
    - Coordenadas del bounding box (x, y, w, h).
    - Clase: La probabilidad de que cada caja delimitadora contenga un objeto específico de las clases posibles (como persona, coche, perro, etc.).
    - Confianza: Una medida de la certeza de la red sobre la predicción.
- La red de YOLO predice un número fijo de celdas en la imagen (por ejemplo, 13x13, 19x19, etc.), y cada celda es responsable de predecir ciertos bounding boxes y sus clases correspondientes.

Capas

La arquitectura de YOLO se basa principalmente en capas convolucionales, que están diseñadas para extraer características espaciales y de texto en las imágenes. Estas capas se componen de varios filtros que se mueven sobre la imagen de entrada.

Capas convolucionales

- **Neuronas convolucionales**: Realizan convoluciones sobre la imagen de entrada. Cada filtro convolucional detecta un patrón específico, como bordes, colores o texturas.
- Estas capas son responsables de aprender representaciones jerárquicas de las características de la imagen.

Capas de activación

- Después de las capas convolucionales, las neuronas pasan a través de una función de activación que introduce no linealidad en la red.

Capas completamente conectadas (Fully Connected - FC)

- En las versiones anteriores de YOLO, como YOLOv1, había una capa completamente conectada que conectaba las salidas de las capas convolucionales a la predicción final. En versiones posteriores (YOLOv3 y YOLOv4), la red tiene una salida que directamente predice los valores sin una capa completamente conectada.

Capas de normalización

Las redes YOLO también usan técnicas como la normalización de lotes (Batch Normalization) para estabilizar el entrenamiento y acelerar la convergencia. Esto ayuda a reducir la dependencia de la inicialización de los parámetros y hace que el modelo sea más robusto.

Neuronas

Neuronas de Entrada

YOLOv5 y YOLO11:

- **Entrada**: La red YOLO toma como entrada una imagen de tamaño fijo, generalmente de 416x416 píxeles o 640x640 píxeles.
    - La entrada a la red será un tensor de 3 dimensiones: (altura, ancho, canales).

        - **Tamaño**: Para una imagen de 640x640 píxeles, se genera un tensor de tamaño (640, 640, 3), donde el 3 representa los canales de color (Rojo, Verde, Azul - RGB).
    - Neuronas de entrada:
        - Cada píxel de la imagen de entrada es considerado como una neurona de entrada. Por lo tanto, para una imagen de 640x640 con 3 canales, tendrás 640 * 640 * 3 = 1,228,800 neuronas de entrada.
    - Tipo de neuronas: Las neuronas de entrada simplemente reciben la información de la imagen y la pasan a través de las capas convolucionales.

Neuronas Ocultas

Las capas ocultas en YOLOv5 y YOLO11 son principalmente capas convolucionales y capas de activación (como ReLU), que realizan el procesamiento y extracción de características de la imagen.

Número de Neuronas Ocultas:

- No hay un número fijo de neuronas ocultas, ya que depende del número de capas convolucionales y la profundidad de la red. En modelos avanzados como YOLOv5 y YOLO11, hay múltiples capas convolucionales, y cada una tiene un número diferente de filtros. Generalmente, el número de filtros aumenta a medida que la red se hace más profunda.
    - YOLOv5 y YOLO11 pueden tener hasta 60 capas convolucionales o más, dependiendo de la versión.

Neuronas de Salida

Capas de Salida (Predicciones)

- En la capa de salida de YOLOv5 y YOLO11, las neuronas predicen las coordenadas de las cajas delimitadoras y las clases de objetos.
- Cada celda de la cuadrícula de la imagen tiene un conjunto de neuronas de salida que predicen:
    - Las coordenadas del bounding box (x, y, ancho, alto).
    - La probabilidad de clase para cada clase de objeto.
    - La confianza (probabilidad de que un objeto esté presente).

Número de Neuronas en la Capa de Salida:

- La cantidad de neuronas de salida depende de:
    - Número de celdas de la cuadrícula: YOLO divide la imagen en una cuadrícula (por ejemplo, 13x13, 19x19). Cada celda tiene 4 predicciones para el bounding box y 1 predicción de confianza.
    - Número de clases: En tu caso, se utilizan 80 clases de objetos (como personas, coches, animales, etc.).

Si tienes una cuadrícula de 13x13 y 80 clases, el número total de predicciones sería:

- (13 * 13) * (4 coordenadas + 1 confianza + 80 clases) = 13 * 13 * 85 = 14,705 neuronas de salida.

Cada una de estas neuronas es responsable de predecir las propiedades mencionadas para cada celda en la cuadrícula de la imagen.

YOLOv5

YOLOv5 es una de las versiones más recientes de la serie YOLO, caracterizada por su velocidad y precisión. Utiliza redes neuronales convolucionales (CNNs) profundas para extraer características de las imágenes y predecir las ubicaciones de los objetos y sus clases en una sola pasada.

YOLO11

YOLO11 es una versión avanzada que mejora las capacidades de YOLOv5 mediante optimizaciones en la arquitectura y el uso de nuevas técnicas de regularización y optimización. Aunque la arquitectura sigue siendo basada en CNNs, YOLO11 introduce mejoras que permiten obtener una mayor precisión en la detección, especialmente en ambientes complejos y con grandes volúmenes de datos.

Las Funciones de Activación

Las funciones de activación son esenciales en las redes neuronales para introducir no linealidad en el modelo y permitirle aprender patrones complejos. Las funciones de activación utilizadas en YOLOv5 y YOLO11 incluyen:

ReLU (Rectified Linear Unit)

Comúnmente utilizada en las capas ocultas de las redes neuronales. La función ReLU permite que el modelo aprenda representaciones no lineales, mejorando su capacidad de generalizar se aplica en las capas convolucionales y en otras capas intermedias.

Expresion matemática:

$$\mathrm{ReLU}(x) = \max(0, x)$$

Sigmoide

La función sigmoide se usa para la predicción de la confianza (probabilidad) de que un objeto esté presente en el bounding box.

Expresion matemática:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Softmax

En algunos casos, se usa para manejar múltiples clases, ya que es útil cuando se trata de clasificación multiclase.

Expresion matemática:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

El Algoritmo de Entrenamiento

El algoritmo de entrenamiento utilizado en YOLOv5 y YOLO11 es un proceso basado en optimización de la función de pérdida mediante retropropagación y el uso de un optimizador como AdamW (una variante de Adam). La función de pérdida utilizada es una combinación de tres términos:

El tamaño del lote de entrenamiento (batch size) fue diferente entre los dos modelos (batch size de 10 para YOLOv5 y batch size de 8 para YOLO11).

Pesos de la Red

Los pesos en una red neuronal son los parámetros que se ajustan durante el entrenamiento. En YOLO, estos pesos se asignan inicialmente de forma aleatoria, y luego se ajustan mediante el proceso de retro propagación utilizando el algoritmo de optimización Adam.

- **Pesos de las capas convolucionales**: Son los filtros que detectan características específicas en las imágenes.

- **Pesos de las capas de salida**: Los pesos en las capas de salida determinan cómo se combinan las características extraídas por las capas convolucionales para generar las predicciones finales.

Entrenamiento y Ajuste de Pesos

Durante el entrenamiento, se actualizan los pesos de la red mediante descenso por gradiente, utilizando un algoritmo de optimización como Adam.

Expresión matemática:

$$\theta_t = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}$$

El algoritmo Adam ajusta la tasa de aprendizaje de cada parámetro de forma adaptativa, utilizando no solo el gradiente actual, sino también un promedio de los gradientes pasados (primer momento) y el cuadrado de los gradientes (segundo momento). Esto le permite manejar eficientemente problemas con grandes datos y parámetros.

El objetivo es minimizar la función de pérdida (loss function), que en el caso de la detección de objetos es una combinación de:

- Error de localización (para las coordenadas del bounding box).
- Error de clasificación (para las clases de objetos).
- Error de confianza (para la certeza de las predicciones).

Los Datos de Entrenamiento

Los datos de entrenamiento utilizados en este proyecto fueron extraídos del conjunto de datos COCO (Common Objects in Context), que contiene imágenes de diversas escenas con objetos etiquetados. Este conjunto de datos es ampliamente utilizado en tareas de detección de objetos y clasificación.

- **Anotaciones**: Las imágenes fueron anotadas con las coordenadas de las cajas delimitadoras que indican la posición y el tamaño de los objetos, y con las clases correspondientes.
- **Preprocesamiento**: Las imágenes fueron redimensionadas (en el caso de YOLO 11) a un tamaño uniforme (por ejemplo, 640x640 píxeles) para ser introducidas en la red neuronal.

Datos de Prueba

El conjunto de datos de prueba consistió en un 20% de las imágenes del conjunto original, que se utilizaron para evaluar el rendimiento de los modelos entrenados.

Imágenes del entramiento en proceso

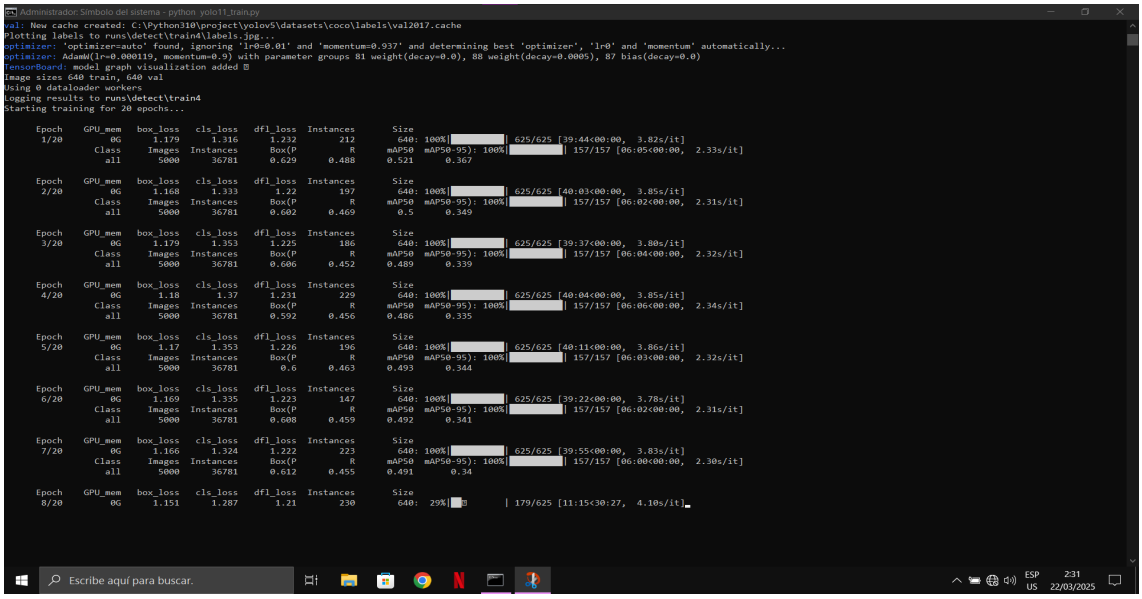

Ilustración 1 Entrenamiento YOLO 11

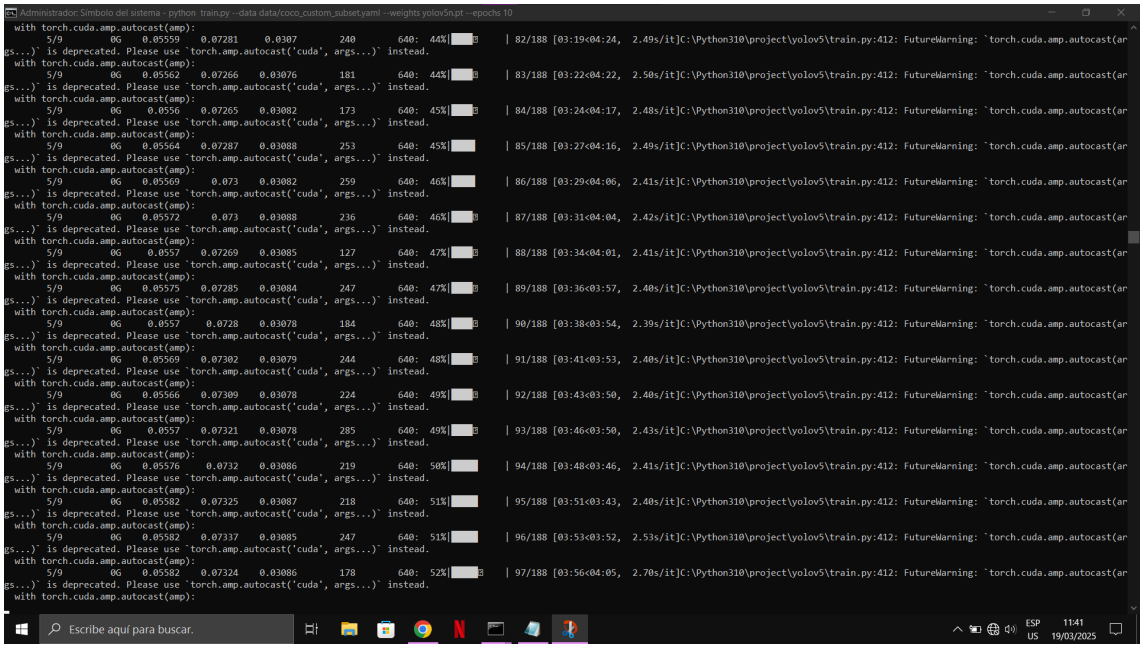

Ilustración 2 Entrenamiento YOLO 5

Resultados

En esta sección se comparan los resultados obtenidos de YOLOv5 y YOLO11 a través de varias métricas.

Matriz de Confusión Normalizada

YOLO11: La matriz de confusión para YOLO11 muestra cómo el modelo ha clasificado cada una de las clases. Puedes observar cómo las clases principales tienen valores más altos, indicando una clasificación correcta.

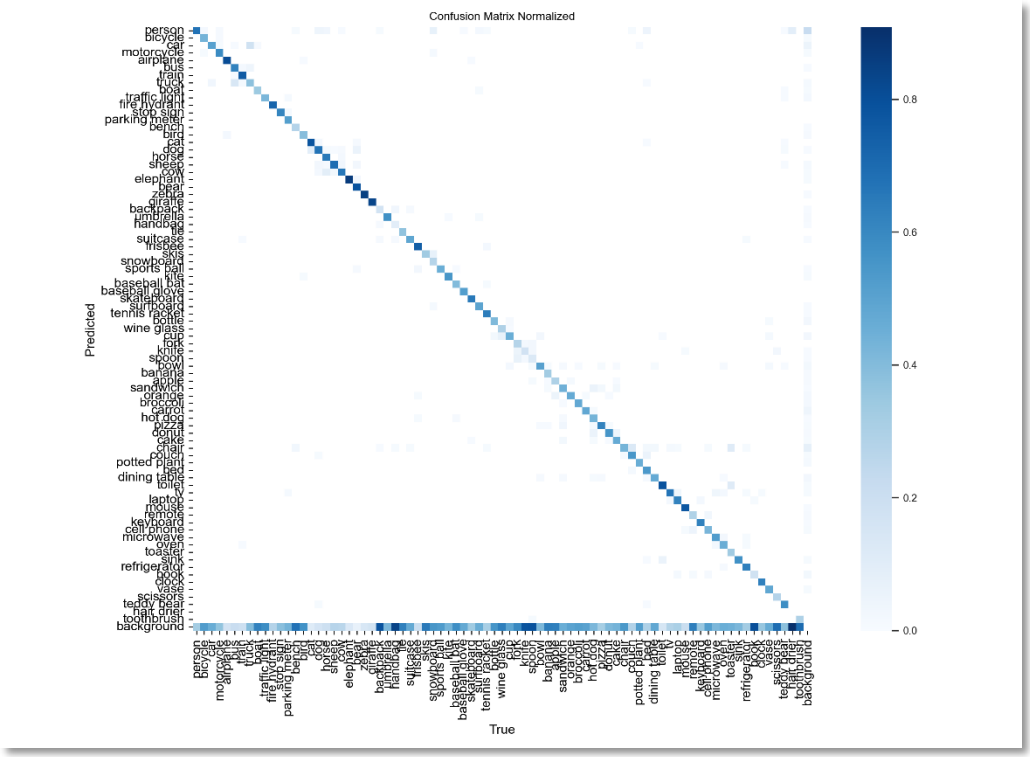

Ilustración 3 Matriz de Confusión Normalizada YOLO 11

YOLOv5: En la matriz de confusión de YOLOv5, se observa un rendimiento menor en algunas clases, lo que sugiere una menor precisión en las predicciones.

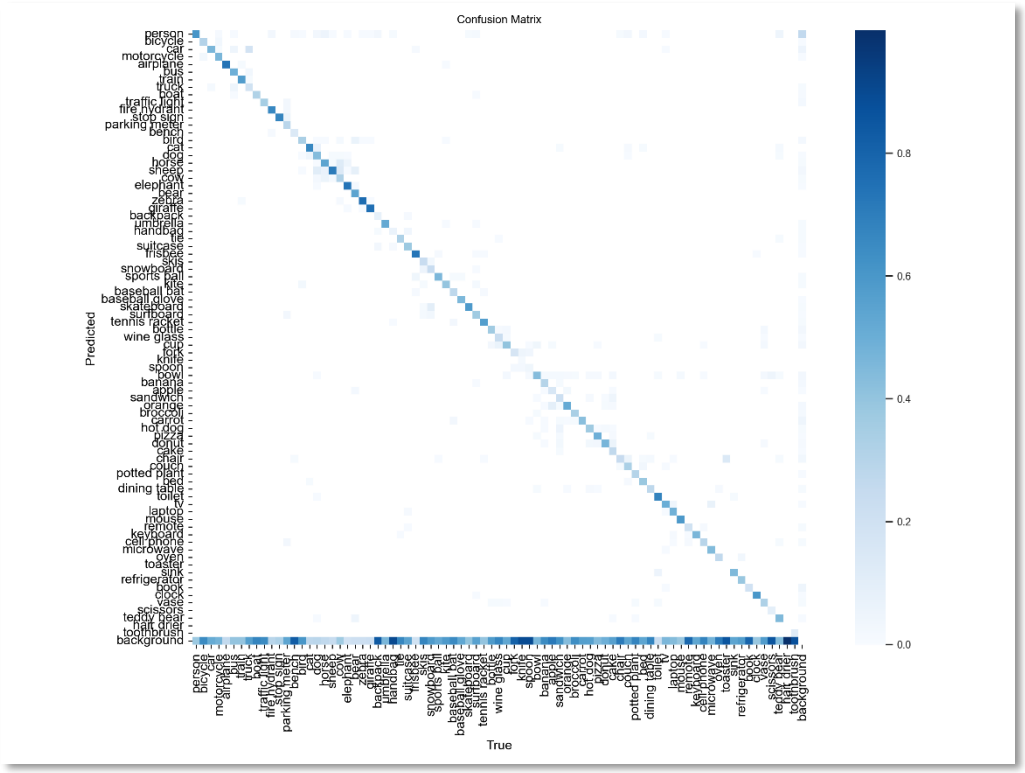

Ilustración 4 Matriz de Confusión Normalizada YOLO 5

Curvas F1-Confidence

YOLO11: En comparación, YOLO11 tiene un F1-Score máximo de 0.53 a un umbral de 0.301, lo que indica un mejor rendimiento en el balance entre precisión y recall.

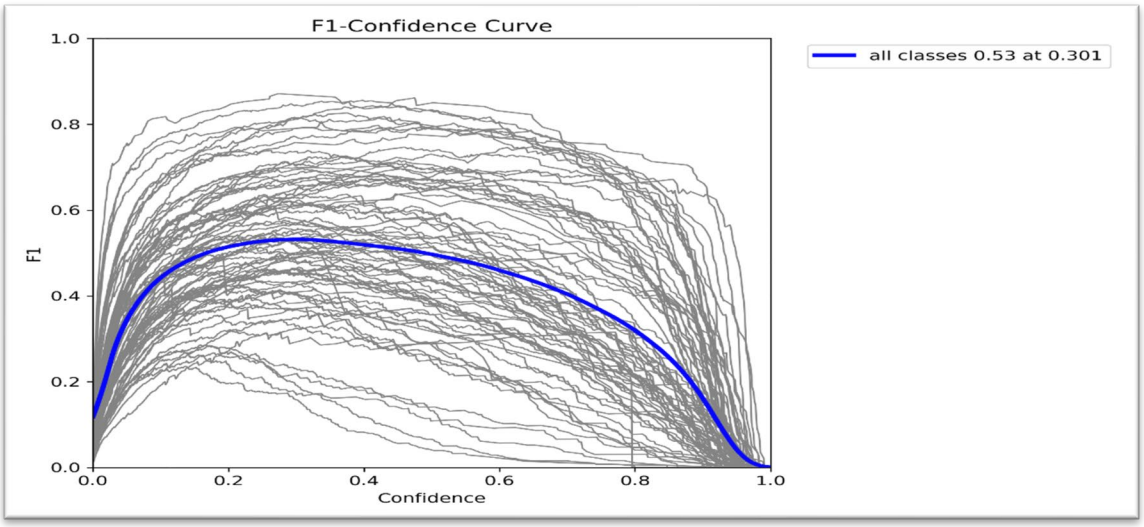

Ilustración 5 Curva F1-Confidence YOLO 11

YOLOv5: La curva de F1-Confidence muestra el equilibrio entre la precisión y la recuperación para diferentes valores de confianza. El modelo tiene un F1-Score máximo de 0.43 a un umbral de 0.257.

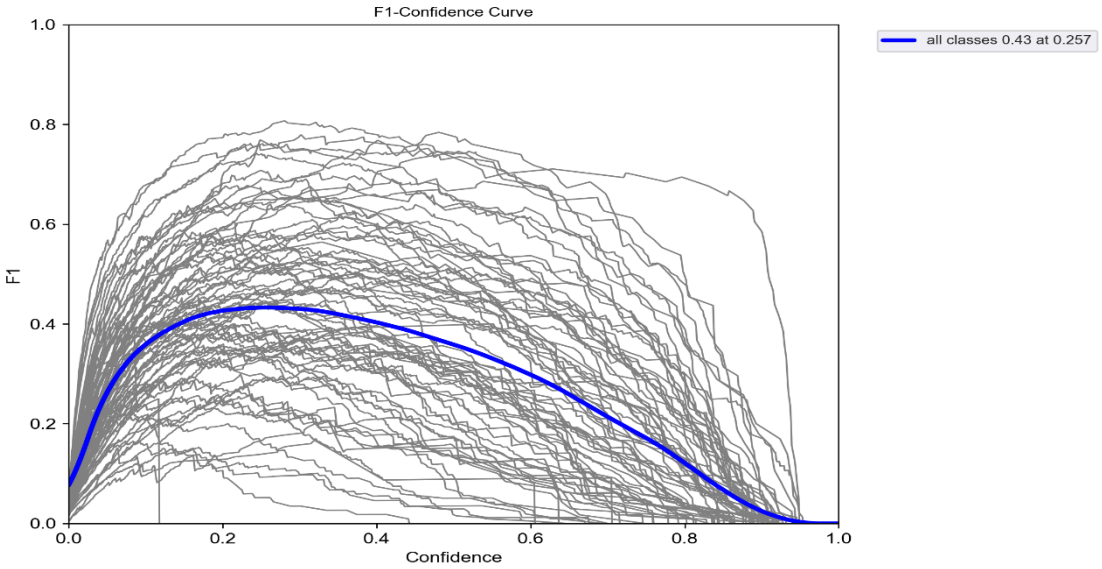

Ilustración 6 Curva F1-Confidence YOLO 5

Curvas Precision-Recall

YOLO11: muestra una mejor precisión y recuperación en el umbral de confianza, lo que le permite tener una mejor capacidad para identificar objetos correctamente.

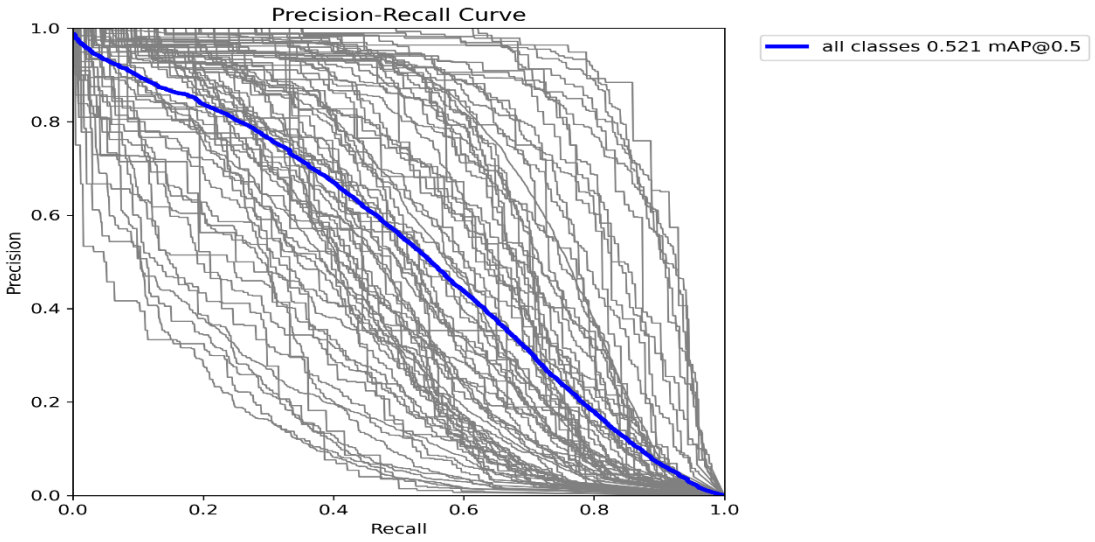

Ilustración 7 Curva Precision-Recall YOLO 11

YOLOv5: La curva de precisión y recall muestra cómo se mantiene la precisión conforme aumenta el recall. YOLOv5 tiene un rendimiento aceptable, pero se observa que puede mejorar en precisión y recuperación.

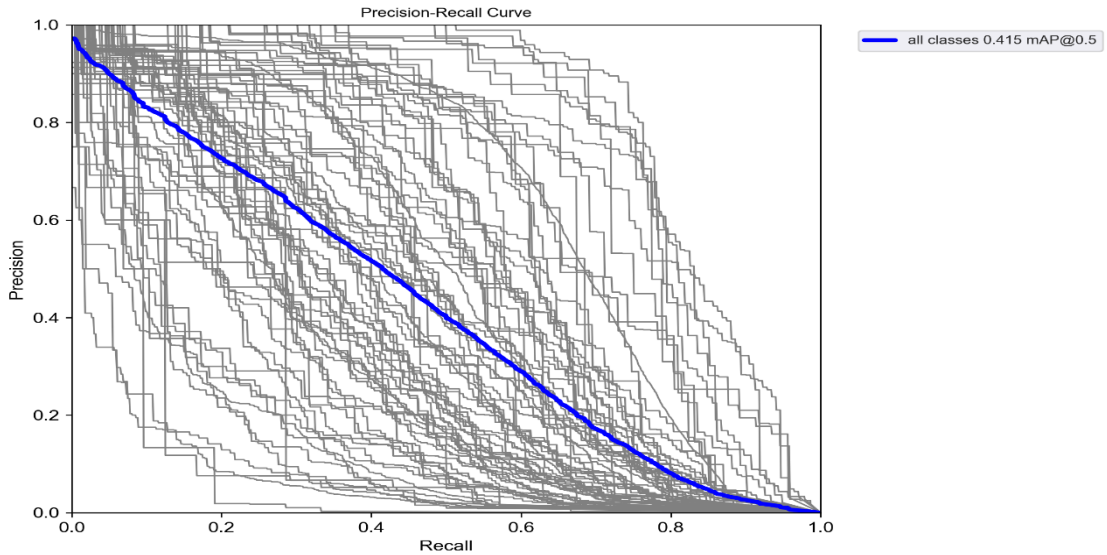

Ilustración 8 Curva Precision-Recall YOLO 5

El gráfico de la curva F1 muestra que, a medida que aumenta el umbral de confianza, el modelo alcanza una mejor precisión, pero esto también reduce el recall. Esto indica que YOLO11 fue más preciso, pero podría haber dejado pasar algunos objetos con baja certeza. En comparación, YOLOv5 alcanzó un mejor recall, pero con menor precisión, lo que implica que identificó más objetos, pero con más falsos positivos.

La matriz de confusión para YOLOv5 muestra que la clase 'person' fue la más correctamente identificada, mientras que otras clases, como 'toilet' y 'refrigerator', tienen más confusión con las clases de fondo. Esto indica que el modelo tuvo dificultades para diferenciar estas clases en ciertas imágenes de prueba.

Comparaciones de imágenes procesadas por ambos modelos

YOLO 11

Ilustración 9 Imagen procesada por YOLO 11

YOLOv5

Ilustración 10 Imagen procesada por YOLO 5

Se puede apreciar que el modelo YOLO 11 puede detectar mas objetos, esto va aunado al entrenamiento más extenso que este llevo nos da mejores reultados.

Discusión sobre las Limitaciones

Limitaciones del Entrenamiento sin GPU

Ambos entrenamientos se realizaron utilizando solo la CPU, lo que afectó considerablemente el tiempo de entrenamiento. Para YOLOv5, el modelo tardó aproximadamente 6 horas en completarse, mientras que YOLO11 tardó casi 24 horas debido a la mayor cantidad de datos y parámetros del modelo.

La falta de una unidad de procesamiento gráfico (GPU) adecuada limitó significativamente el rendimiento de los modelos, aumentando el tiempo de entrenamiento. Este factor contribuyó a que, a pesar de tener más datos y un mayor número de épocas, los resultados no fueran tan óptimos como podrían ser con el uso de una GPU potente.

Conclusiones

- YOLO11 demostró ser un modelo superior en términos de precisión, recuperación y F1-Score en comparación con YOLOv5. Sin embargo, el entrenamiento sin el uso de una GPU afectó los tiempos de entrenamiento, siendo significativamente más largo en YOLO11.

- Ambos modelos, sin embargo, demostraron ser efectivos para la detección de objetos en imágenes, y se recomienda considerar el uso de una GPU para acelerar el proceso de entrenamiento y mejorar el rendimiento general de los modelos.

- Para mejores resultados se necesitaría un entramiento mucho más extenso, esto sin lugar a duda proporcionaría datos más exactos y mejores resultados.

- Si bien se quiere llevar a una comparación de modelos como tal no podemos decir objetivamente que YOLOv5 es inferior a YOLO 11 porque se probó con parámetros distintos, mas sin embargo los datos oficiales del modelo dan por mejor a YOLO 11

Recomendaciones

- **Uso de GPU**: Se recomienda encarecidamente entrenar ambos modelos con una GPU para mejorar los tiempos de entrenamiento y la eficiencia general.

- **Parámetros de Entrenamiento**: Es recomendable experimentar con diferentes configuraciones de parámetros, especialmente el tamaño del lote (batch size) y el tamaño de imagen (imgsz), para encontrar la combinación que balancee el tiempo de entrenamiento y la precisión. Además, el uso de fine-tuning (ajuste fino) en modelos pre-entrenados podría acelerar el proceso de entrenamiento y mejorar los resultados con menos datos.

- **Mejores Resultado**s: Una estrategia para mejorar los resultados sería aumentar el tamaño del conjunto de datos con más diversidad (por ejemplo, diferentes tipos de imágenes o condiciones de iluminación) y emplear data augmentation para reducir el sobreajuste y mejorar la capacidad general del modelo.

- **Uso de otros modelos**: Aquí se hicieron las pruebas con YOLO tanto la versión cinco de este y la versión once sin embargo estos no son los únicos modelos y tampoco los mejores, para obtener un modelo más eficiente se debería hacer pruebas con otros modelos como ser RF-DETR o MobileNet.