Universidad Nacional Autónoma de México

Facultad de Ciencias

Inteligencia Artificial

Proyecto final de detección de automóviles usando modelos YOLO

Alumnos:
Acevedo Romero Miroslava
Martínez Cano Ricardo Iván
Victoria Morales Ricardo Maximiliano

Profesor:
Sergio Rodolfo Cruz Gómez

Grupo:
7001

Ciclo escolar:
2025-2

● Roles y Contribuciones

Acevedo Romero Miroslava

Rol: Evaluación y análisis de resultados

Contribución: Planteamiento del problema aplicativo, definiendo los objetivos, y entrenamiento de una red neuronal que procesa el dataset de automóviles, también se encargó del análisis de los resultados finales para el problema planteado.

Martínez Cano Ricardo Iván

Rol: Especialista en preprocesamiento de datos y diseño de la red neuronal

Contribución: Responsable del preprocesamiento de los datos y el diseño de la red neuronal. También se encargó del análisis de los resultados de los diferentes modelos utilizados.

Victoria Morales Ricardo Maximiliano

Rol: Project Manager

Contribución: Se encargó de agendar reuniones con el equipo para hacer un plan de trabajo y coordinar los avances del proyecto. También buscó los datasets adecuados para el problema en cuestión y propuso los diferentes modelos de CNN para el entrenamiento de la red.

● Descripción del problema

La detección de objetos a partir de fragmentos de video tiene entre sus aplicaciones mejorar la seguridad vial y facilitar la identificación de señas particulares. Con esto en mente, nos enfocamos en la detección de automóviles con fines de seguridad, como pueden ser las fotomultas, la búsqueda de automóviles y un mejor manejo de reportes por robo.
Habiendo más necesidad por asegurarnos de que los sistemas automáticos de seguridad sean confiables, buscamos crear un modelo que detecte autos con precisión a partir de videos tomados por cámaras de seguridad y en las carreteras.

- **Requerimientos específicos:**
    - **Rápido procesamiento:** teniendo en mente aplicaciones en tiempo real (como en los sistemas de fotomultas), buscamos que el sistema reconozca automóviles con rapidez.
    - **Procesamiento de imágenes de alta resolución:** en caso de que sea necesario identificar placas, es necesario que pueda identificarse al automóvil, minimizando confusiones y ambigüedades.
    - **Precisión alta:** relacionado con el objetivo anterior, es necesario que los automóviles puedan detectarse, minimizando sobre todo los falsos negativos (no detectar un automóvil en curso).

- **Contexto de aplicación:** el sistema puede aplicarse en espacios como carreteras, casetas de vigilancia y sistemas de seguridad vial en general.

Base de datos y preprocesamiento

Descripción
Para entrenar un modelo YOLO que sea capaz de detectar autos en fotos, seleccionamos un conjunto de datos de la plataforma Kaggle. El nombre de éste conjunto de datos es “Car Object Detection” y está disponible en:
https://www.kaggle.com/datasets/sshikamaru/car-object-detection/data

Al descargar la carpeta comprimida obtenemos 2 carpetas ("testing_images" y "training images") con fotos de carreteras con y sin carros.

También obtenemos dos archivos del tipo Comma Separated Values “sample_submission.csv” y “train_solution_bounding_boxes (1).csv”. El primero funge como una plantilla de envío, no contiene las etiquetas reales. El segundo sí contiene las etiquetas reales y es el que se usa.

La carpeta “training images” sirve como los datos para el entrenamiento. A continuación un vistazo de su contenido:

La carpeta "testing_images" tiene fotografías similares.

El archivo “train_solution_bounding_boxes (1).csv” contiene en cada línea un nombre de un archivo de imágen de las carpetas anteriores, y las coordenadas de la ubicación del auto en la fotografía (cajas delimitadoras). Así se ven las primeras líneas:

proyecto-final-inteligencia-artificial > data > train_solution_bounding_boxes (1).csv

```
1   image,xmin,ymin,xmax,ymax
2   vid_4_1000.jpg,281.2590449,187.0350708,327.7279305,223.225547
3   vid_4_10000.jpg,15.16353111,187.0350708,120.3299566,236.4301802
4   vid_4_10040.jpg,239.1924747,176.7648005,361.9681621,236.4301802
5   vid_4_10020.jpg,496.4833575,172.3632561,630.0202605,231.5395753
6   vid_4_10060.jpg,16.63096961,186.5460103,132.5586107,238.3864221
7   vid_4_10100.jpg,447.568741,160.6258044,582.0839363,232.5176963
```

Justificación
Este conjunto de datos fue diseñado para la detección de objetos, específicamente de automóviles. Además incluye 1001 fotos para el conjunto de entrenamiento y 175 para el de validación, lo que es suficiente para entrenar a una red neuronal convolucional YOLO. Así mismo, es fácil la transformación de estos archivos a entradas para nuestra red neuronal.

Sus fotografías son variadas, algunas con más de un carro, y en diferentes lugares y desde diferentes perspectivas. En general varía el contenido, la iluminación y la complejidad de sus elementos. Lo que nos ayuda a obtener una red neuronal robusta.

Características
Como se mencionó anteriormente, las imágenes contienen una gran diversidad escénica. Las fotografías son archivos de extensión .jpg, tienen un tamaño de 100 kb aproximadamente, tienen dimensiones 676 x 380, una resolución horizontal y vertical de 96 ppp, y una profundidad de 24 bits.

Procesamiento
Debido a que el conjunto de datos fue hecho específicamente para YOLO, no agregamos un procesamiento adicional.

● Modelos de CNN usados

Se utilizaron tres versiones de la arquitectura de YOLO (You Only Look Once) para resolver el problema de la detección de automóviles.

- YOLOv5u
    - Su arquitectura se origina del modelo YOLOv5 desarrollado por Ultralytics pero integra un modelo multi cabeza libre de anclas y libre objetividad introducido previamente en YOLOv8. Mejora la relación velocidad-precisión en las tareas de detección, ofreciendo una solución robusta para aplicaciones de investigación y de práctica.
    - Reportó una precisión del 96% y un recall del 97% presentando una eficiencia bastante mejorada en comparación al modelo YOLOv5 original. Tarda más tiempo en ejecutarse que YOLOv8, pero vemos una mejora en el porcentaje del recall por lo que tenemos una disminución en los falsos negativos, lo que nos sirve para detectar más elementos correctamente.
- YOLOv8
    - Fue un modelo lanzado en enero de 2023 que ofrece un rendimiento mejorado para precisión y velocidad. Introduce nuevas mejoras y optimizaciones que lo convierten en una opción ideal para la extracción de características y la detección de objetos.

    - Reportó una precisión del 96% y un recall del 95%, así como una velocidad de 88.9 ms, mostrando una detección más veloz que los demás modelos sin abandonar la eficiencia.
- YOLOv11
    - Es el último modelo desarrollado por Ultralytics, ofreciendo detectores de objetos con precisión, velocidad y eficiencia innovadores. Introduciendo gran cantidad de mejoras en la arquitectura y métodos de entrenamiento. Ofrece mejor precisión con menos parámetros, así como una alta adaptabilidad a través de distintos ecosistemas, permitiendo su uso en una gran cantidad de tareas.
    - Reportó una precisión del 96% y un recall del 94% presentando una buena eficiencia y una buena velocidad. Es el modelo más reciente, sin embargo los resultados en esta ocasión no muestran una diferencia considerable respecto a los demás modelos.

Elegimos estos modelos para poder comparar las distintas versiones de YOLO y conocer cómo se comparan al momento de resolver el problema de detección de vehículos. Utilizamos el modelo YOLOv5u ya que es una versión actualizada de un modelo anterior, lo que nos dará una perspectiva de que tanto se ha mejorado este modelo. YOLOv8 es uno de los modelos más innovadores presentados por Ultralytics, por lo que consideramos interesante ver cómo se comparaba a los demás modelos. YOLOv11 es el modelo más reciente de esta arquitectura, por lo que debería presentar las mayores mejoras a la arquitectura hasta ahora.

● Métricas usadas

- **mean Average Precision (mAP)**. Es la métrica más completa para evaluar la calidad general del detector. Captura la precisión del modelo en varios niveles de exigencia para la superposición entre predicción y verdad (IoU: *Intersection over Union*). Equivale al promedio de la métrica Precisión Promedio en todas las clases de un modelo. Se puede usar mAP para comparar tanto modelos diferentes en la misma tarea como versiones diferentes del mismo modelo. mAP se mide entre 0 y 1.

- **Precisión**. Evalúa la capacidad de un modelo para predecir correctamente los casos positivos, es decir, mide cuántas de las predicciones positivas fueron correctas. Se calcula dividiendo el número de verdaderos positivos (VP) entre el total de predicciones positivas (VP + FP).

$$\text{Precision} = \frac{\text{correctly classified actual positives}}{\text{everything classified as positive}} = \frac{TP}{TP + FP}$$

- **Recall**. Mide la capacidad de un modelo para identificar correctamente todos los casos positivos de una clase.

$$\text{Recall (or TPR)} = \frac{\text{correctly classified actual positives}}{\text{all actual positives}} = \frac{TP}{TP + FN}$$

Se puede observar en la siguiente celda un poco de cómo obtuvimos las métricas en nuestro proyecto:

```python
# Evaluar el modelo
model = YOLO("runs/yolov8n_custom/weights/best.pt")
metrics = model.val()

# Mostrar las métricas
print(f"mAP@.5:.95 = {metrics.box.map:.4f}")
print(f"mAP@.50    = {metrics.box.map50:.4f}")
print(f"mAP@.75    = {metrics.box.map75:.4f}")

print(f"Precision   = {metrics.box.mp:.4f}")
print(f"Recall      = {metrics.box.mr:.4f}")
```

Las métricas se obtienen a través de “model.val()”, éstas corresponden a las métricas estándar para modelos de detección.

```python
print(f"mAP@.5:.95 = {metrics.box.map:.4f}")
```

imprime la mAP para distintos umbrales de intersección sobre unión, desde 0.5 hasta 0.95 en pasos de 0.05.

```python
print(f"mAP@.50    = {metrics.box.map50:.4f}")
```

Imprime la mAP para un umbral de intersección sobre unión de 0.5.

```python
print(f"mAP@.75    = {metrics.box.map75:.4f}")
```

Imprime la mAP para un umbral de intersección sobre unión de 0.75.

```python
print(f"Precision   = {metrics.box.mp:.4f}")
print(f"Recall      = {metrics.box.mr:.4f}")
```

imprimen la precisión y el recall respectivamente.

En una ejecución, la celda devolvió lo siguiente:

```
Results saved to runs\detect\val2
mAP@.5:.95 = 0.6587
mAP@.50    = 0.9843
mAP@.75    = 0.7826
Precision   = 0.9647
Recall      = 0.9417
```

`mAP@.5:.95 = 0.6587` indica que el modelo tiene un buen rendimiento promedio, `mAP@.50 = 0.9843` que detecta correctamente casi todos los objetos, `mAP@.75 = 0.7826` indica que también tiene un buen rendimiento con mayor exigencia para localizar correctamente, `Precisión = 0.9647` que de las detecciones hechas casi todas fueron correctas, y `Recall = 0.9417` indica que detectó a casi todos los autos que debía detectar.

Razonamiento para la selección
En un proyecto para la detección de carros con YOLO, las métricas de precisión, recall y mAP son muy importantes para analizar qué tan bien funciona el modelo.

La precisión indica cuántas de las detecciones fueron correctas. De esta forma podemos enterarnos de cuántos de los carros detectados realmente lo eran. El recall nos dice cuántos carros reales detectó el modelo, para tratar de detectar la mayor cantidad posible. La mAP combina la precisión y el recall y evalúa qué tan bien localiza el modelo los autos en distintos niveles de exigencia.

Gracias a las métricas anteriores podemos saber si el modelo funciona de manera confiable y cómo podemos mejorarlo.

En la siguiente sección, se compararán las métricas con más modelos.

- ## Resultados obtenidos (incluye Análisis y comparación de resultados)

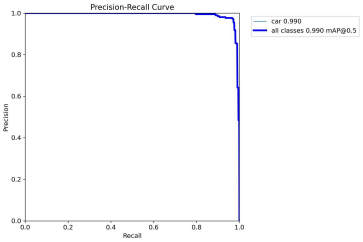
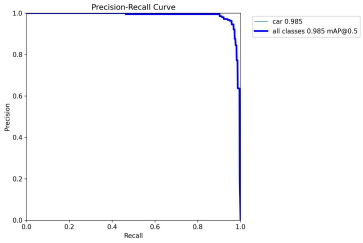
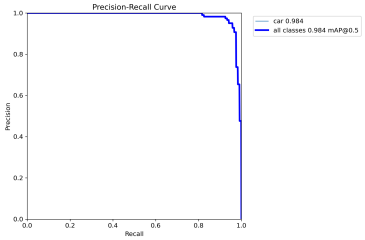
A continuación se presenta una tabla de comparación de resultados de los modelos:

Modelo	Precisión	Recall	Average Precision (mAP)	Velocidad
YOLOv5u	96.490%	97.722%	99%	109.244ms
YOLOv8	96.270%	95.560%	98.5%	88.9ms
YOLOv11	95.851%	96.283%	98.396%	102.2802ms

Análisis de resultados

- YOLOv5u: El modelo cuenta con la precisión y recall más alto entre todos los modelos utilizados, sin embargo también es el modelo que tarda más tiempo en predecir los objetos en las imágenes. Sin embargo, su eficacia es mayor a la de los demás modelos comparados, con mayor precisión promedio y recall, lo que indica que detecta apropiadamente la mayor cantidad de vehículos.
- YOLOv8: El modelo es el que toma menos tiempo para obtener resultados, tiene buen nivel de precisión, pero el recall es el más bajo entre los modelos comparados, por lo que hay un mayor número de falsos negativos. Esto muestra una mayor cantidad de resultados positivos que el modelo no puede encontrar.
- YOLOv11: El modelo presenta la menor precisión entre todos los modelos comparados, por lo que detecta varios falsos positivos. Sin embargo su recall no es tan bajo por lo que se puede utilizar si buscas disminuir la cantidad de falsos negativos.

A continuación, mostramos las gráficas Precisión - Recall, las cuales nos muestran la relación entre la precisión del modelo y el recall para los distintos modelos utilizados, aquí vemos que el modelo con el mayor área bajo la curva es YOLOv5u, lo que nos indica que este es el modelo que mejor detecta los vehículos y el que tiene la mayor eficiencia para solucionar el problema presentado.

Curva Precisión-Recall YOLOv5u	Curva Precisión-Recall YOLOv8	Curva Precisión-Recall YOLOv11
		

● Análisis final y conclusiones

- **Modelo recomendado:** con base en los análisis de las métricas, concluimos que el mejor modelo para nuestros objetivos funciona con **YOLOv8**, pues aunque no tiene la mayor precisión de los modelos, **YOLOv8** resulta mucho más rápido para procesar los mismos datos. Si el sistema prioriza un procesamiento más inmediato, como en un sistema de fotomultas, esta puede ser una mejor opción. Además, la diferencia de precisión no es tan grande con modelos como **YOLOv5u.**

- **Análisis de contexto de uso:** Para aplicaciones que deban funcionar en tiempo real o con vehículos que pasan a alta velocidad, resulta mejor el modelo con **YOLOv8**, aunque seguramente su buen funcionamiento dependa mucho del hardware con que se opere.

- **Recomendaciones para futuros trabajos:** Para futuros estudios se recomienda usar técnicas con el objetivo de combinar la precisión de algunos modelos con la rapidez de otros, así como probar el modelo en diferentes equipos para comparar su rapidez, así como hacer un entrenamiento con datasets más específicos o buscando detectar caracteres en placas de automóviles, lo cual también podría combinarse con aplicaciones de PLN.