DESARROLLO DE UN SISTEMA INTELIGENTE DE AFORO VEHICULAR EN TIEMPO REAL MEDIANTE RECONOCIMIENTO DE IMÁGENES Y REDES NEURONALES

BRANDON GIOVANNY SANCHEZ CASTELLANOS
PEDRO FERNANDO BRAVO VALLEJOS

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD TECNOLÓGICA
INGENIERÍA CIVIL
BOGOTÁ
2024

DESARROLLO DE UN SISTEMA INTELIGENTE DE AFORO VEHICULAR EN TIEMPO REAL MEDIANTE RECONOCIMIENTO DE IMÁGENES Y REDES NEURONALES

BRANDON GIOVANNY SANCHEZ CASTELLANOS
PEDRO FERNANDO BRAVO VALLEJOS

TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE INGENIERO CIVIL

EDGAR ORLANDO LADINO MORENO, INGENIERO CIVIL, INGENIERO TOPOGRÁFICO MAGÍSTER EN INGENIERÍA CIVIL

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD TECNOLÓGICA
INGENIERÍA CIVIL
BOGOTÁ
2024

Nota de Aceptación

_____

_____

_____

_____

JURADO M.Sc. EDGAR ORLANDO LADINO

_____

JURADO ING. RODRIGO ESQUIVEL RAMÍREZ

BOGOTÁ __ DE JULIO DEL 2024

AGRADECIMIENTOS

Deseamos expresar nuestra más profunda gratitud a todas las personas que nos han brindado su apoyo incansable a lo largo de este proceso.

En primer lugar, queremos agradecer especialmente al M.Sc. Edgar Orlando Ladino, nuestro director de trabajo de grado. Su extraordinaria orientación y su enfoque innovador nos inspiraron a alcanzar niveles de excelencia que inicialmente parecían inalcanzables. Su guía nos permitió ampliar nuestros horizontes y lograr resultados que superaron nuestras expectativas iniciales.

Extendemos nuestro reconocimiento a la Universidad Distrital Francisco José de Caldas por proporcionarnos las herramientas y habilidades necesarias para destacar en cualquier ámbito profesional. La formación recibida ha sido fundamental para nuestro crecimiento académico y personal.

A todos ustedes, nuestro más sentido agradecimiento por ser parte esencial de este viaje académico y contribuir significativamente a nuestro éxito.

DEDICATORIA

Dedico esta Tesis a mi madre, Lilia, cuyo amor incondicional ha sido mi faro en este largo camino. Sus dulces consejos y apoyo inquebrantable han sido fundamentales para alcanzar esta meta.

A mis hermanos, Mishell y Julian, quienes son mi gran fuente de inspiración y un ejemplo constante de perseverancia y excelencia.

A mi novia Hellen, por absolutamente todo. Su comprensión, paciencia y amor han sido mi fortaleza en cada paso de este viaje académico.

A mi padre Marco, por sus valiosos consejos y su actitud positiva frente a los grandes retos. Su sabiduría y optimismo han sido una guía invaluable en mi formación profesional y personal.

A todos ustedes, pilares fundamentales de mi vida, dedico este logro con profunda gratitud y amor.

Brandon Giovanny Sánchez Castellanos

Dedico este trabajo a mis padres, Rocío y Pedro, por su amor incondicional, sacrificio y apoyo constante. Ustedes han sido la fuerza que me impulsa a superar cualquier obstáculo y a perseguir cada uno de mis sueños.

A mi hermano Ángel, por ser una motivación constante para superarme en cada momento de mi vida.

A mis amigos y compañeros de estudio, por los momentos compartidos, las risas y el apoyo mutuo que hicieron más llevadero este desafío.

Este trabajo es el fruto de mi esfuerzo, pero también del amor y apoyo de todos ustedes. Gracias por creer en mí y acompañarme en este viaje.

Pedro Fernando Bravo Vallejos

TABLA DE CONTENIDO

1. RESUMEN **12**
2. ABSTRACT **13**
3. IDENTIFICACIÓN DEL PROBLEMA **14**
4. JUSTIFICACIÓN **15**
5. OBJETIVOS **16**
5.1. OBJETIVO GENERAL 16
5.2. OBJETIVOS ESPECÍFICOS 16
6. MARCO TEÓRICO **17**
6.1. INTELIGENCIA ARTIFICIAL 17
6.2. VISIÓN POR COMPUTADOR 17
6.3. APRENDIZAJE PROFUNDO EN LA VISIÓN POR COMPUTADOR 17
6.4. FUNCIÓN DE PÉRDIDA 18
6.5. REDES NEURONALES ARTIFICIALES (RNA) 18
6.6. FEEDFORWARD NEURAL NETWORKS (FNN) RED NEURONAL DE PROPAGACIÓN HACIA ADELANTE 19
6.7. DEEP BELIEF NETWORKS (DBN) RED DE CREENCIA PROFUNDA 19
6.8. RECURRENT NEURAL NETWORK (RNN) RED NEURONAL RECURRENTE 20
6.9. CONVOLUTIONAL NEURAL NETWORKS (CNN) RED NEURONAL CONVOLUCIONAL 20
6.10. CONTEO DE MULTITUDES Ó CROWD COUNTING 21
6.11. ESTIMACIÓN DE LA DENSIDAD 22
6.12. HAAR CASCADE 22
6.13. ARQUITECTURA DE CLASIFICADOR EN CASCADA 22
6.14. YOU ONLY LOOK ONCE (YOLO) 23
6.15. BOUNDING BOX (CUADRO DELIMITADOR) 23
6.16. ARQUITECTURA DE YOLO 23
6.17. LOCALIZATION LOSS (PÉRDIDA DE LOCALIZACIÓN) 24
6.18. CONFIDENCE LOSS PÉRDIDA DE CONFIANZA 24
6.19. SINGLE SHOT DETECTOR (SSD) 24
6.20. SEGMENT ANYTHING MODEL (SAM) 24
6.21. C2F (CROSS-STAGE PARTIAL BOTTLENECK WITH TWO CONVOLUTIONS) 25
6.22. CPU (UNIDAD CENTRAL DE PROCESAMIENTO): 25
6.23. GPU (UNIDAD DE PROCESAMIENTO GRÁFICO): 26

7. MATERIALES **27**
7.1. XIAOMI NOTE 8 PRO 27
7.2. TRIPODE EN ALUMINIO: 28
7.3. CABLE USB TIPO C 29
8. MÉTODOS **30**
8.1. ADQUISICIÓN DE DATOS 30
8.2. PROCESAMIENTO DE DATOS 34
8.2.1. Cargar los Datos al Computador 35
8.2.2. Adobe Express 35
8.2.3. Google Drive 36
8.2.4. Google Collaboratory (Colab) 36
8.2.5. Etiquetado de Imágenes 37
8.2.6. Revisión de las Imágenes Etiquetadas por Medio de la Herramienta ROBOFLOW 37
8.2.7. Aumento de Imágenes (Data Augmentation) 38
8.2.8. Redimensionamiento de Imágenes 39
8.3. ENTRENAMIENTO DEL MODELO 39
8.3.1. Imágenes de Entrenamiento, Validación y Testeo 40
8.3.2. Parámetros del Modelo 41
8.3.3. Número de Épocas (Epochs) 42
8.3.4. Tamaño de Lote (Batch Size) 42
8.3.5. Elección del Modelo 43
8.3.5.1. YOLOv8n 45
8.3.5.2. YOLOv8s 45
8.3.5.3. YOLOv8m 45
8.3.5.4. YOLOv8l 46
8.3.5.5. YOLOv8x 46
8.3.6. Redimensionamiento 47
8.3.7. Normalización 47
8.3.8. Entorno de Ejecución 48
8.4. ARQUITECTURA DE LA RED NEURONAL CONVOLUCIONAL (RNC) 51
8.4.1. Backbone (YOLOv8 CSPDarknet) 53
8.4.2. Neck (YOLOv8PAFPN) 53
8.4.3. Head (YOLOv8HeadModule) 54
8.4.4. Función de Activación 55
8.4.5. Compilación del Modelo 56

8.5. ESTADÍSTICOS DE CONTROL 57
8.5.1. Precisión(P) y Recall(R) 58
8.5.2. Mean Average Precision (mAP) 59
8.5.3. F1 Score 60
8.5.4. Confianza (Confidence) 61
8.5.5. Matriz de Confusión 61
8.5.6. Losses: Box Loss, Cls Loss, Dfl Loss 63
8.5.7. Velocidades de Procesamiento 64
8.5.8. Entropía Cruzada Categórica Dispersa 65
9. RESULTADOS Y DISCUSIÓN 67
9.1. MÉTRICAS DE EVALUACIÓN 69
9.1.1. Curvas de pérdida (loss) 69
Curvas de pérdida (loss) 69
Box Loss: 69
Cls Loss: 70
Dfl Loss: 70
Precisión: 70
Recall: 71
mAP50 y mAP50-95: 71
9.2. ANÁLISIS VISUAL DE LAS GRÁFICAS 72
9.2.1. Recall-Confidence Curve 72
Recall-Confidence Curve 72
9.2.2. F1-Confidence Curve 73
F1-Confidence Curve 73
9.2.3. Precision-Confidence Curve 74
Precision-Confidence Curve 74
9.2.4. Precision-Recall Curve 75
Precision-Recall Curve 75
9.2.5. Matriz de Confusión 76
Matriz de Confusión 76
Carro:
Verdaderos Positivos (TP): La celda de la diagonal principal (Carro-Carro) muestra un valor de 0.99, indicando que el modelo tiene una alta precisión en la detección de carros. 77
Moto:
Verdaderos Positivos (TP): La celda de la diagonal principal (Moto-Moto)

muestra un valor de 0.98, lo que indica una alta precisión en la detección de motos. 77
Bus/Camión: 77
Fondo (Background): 78
9.3. VALIDACIÓN 78
9.3.1. Video Procesado 78
Código 1: Detección, Clasificación y Conteo de Vehículos con Generación de Gráficas. 79
Código 2: Cálculo de velocidad para cada clase con Generación de Gráficas. 81
9.3.2. Tiempo Real 84
Código 3: Detección Precisa de Vehículos con YOLOv8 85
Código 4: Detección, Clasificación y Conteo de Vehículos en Tiempo Real con Generación de Gráficas en Tiempo Real: 86
9.4. Predicción 89
Video Procesado 89
Código 1 y 2: Detección, Clasificación, Conteo y velocidad promedio de Vehículos con Generación de Gráficas. 89
Tiempo Real 93
Código 3: Detección Precisa de Vehículos con YOLO 93
Código 4: Detección, Clasificación y Conteo de Vehículos en Tiempo Real con Generación de Gráficas en tiempo real:
1.0 CONCLUSIONES 94
11. RECOMENDACIONES 98
General: 98
Recomendaciones Técnicas: 98
Recomendaciones de Captura y Calidad de Datos: 99
Recomendaciones de Gestión y Organización: 100
Errores Comunes a Evitar: 100
12. REFERENCIAS 102
106

TABLA DE FIGURAS

Figura 1 *Teléfono Móvil Redmi Note 8 Pro*, s. f. 28
Figura 2 *Trípode.* 29
Figura 3 *Ubicación Toma de Datos.* 30
Figura 4 *Toma de Datos.* 31
Figura 5 *Puntos de Referencia.* 33
Figura 6 *Etiquetado de imágenes en la plataforma roboflow.* 38
Figura 7 *Ubicación Toma de Datos.* 30
Figura 8 *Imágenes de validación.* 41
Figura 9 *Diferentes modelos de YOLOv8.* 44
Figura 10 *GPU NVIDIA T4* 50
Figura 11 *Arquitectura detallada de YOLOv8* 52
Figura 12 *Arquitectura de YOLOv8.* 55
Figura 13 *NMS (Supresión de no máximos).* 57
Figura 14 *Precisión Promedio(AP) e Intersección sobre Unión (IoU).* 60
Figura 15 *Matriz de Confu*sión. 63
Figura 16 *Interfaz de RoboFlow.* 68
Figura 17 *Curvas de pérdida (loss).* 69
Figura 18 *Recall-Confidence Curve.* 72
Figura 19 *F1-Confidence Curve.* 73
Figura 20 *Precision-Confidence.* 74
Figura 21 *Precision-Recall Curve.* 75
Figura 22 *Matriz de Confusión.* 76
Figura 23 *Resultados código 1.* 80
Figura 24 *Diagrama de barras código 1.* 81
Figura 25 *Resultados código 2.* 83
Figura 26 *Diagrama de barras código 2.* 84
Figura 27 *Resultados código 3.* 86
Figura 28 *Resultados código 4.* 87
Figura 29 *Diagrama de barras código 4.* 88

1. RESUMEN

La obtención de datos precisos sobre el flujo vehicular en tiempo real es indispensable para la gestión y optimización de las redes de transporte. Los métodos actuales en Bogotá, como el conteo manual y los sensores fijos, presentan limitaciones significativas en cobertura y precisión. Este proyecto aborda estos desafíos mediante el desarrollo de un sistema inteligente de aforo vehicular en tiempo real utilizando redes neuronales convolucionales y visión por computador. Utilizando el modelo YOLOv8n, se realizó la detección y clasificación de vehículos en tres categorías: carros, motos y buses/camiones.

Se capturaron videos de alta resolución en un punto estratégico de Bogotá, y se procesaron las imágenes para entrenar el modelo. Las técnicas de procesamiento de datos incluyeron etiquetado, aumento y normalización de imágenes. Las métricas de evaluación utilizadas, como precisión, recall, mAP, F1 Score, y análisis de la matriz de confusión, demostraron la alta efectividad del modelo en la detección vehicular.

Además, se discutieron las velocidades de procesamiento y la eficiencia del sistema, cruciales para aplicaciones en tiempo real. Los resultados reflejan un alto nivel de precisión y recall, sugiriendo la viabilidad del sistema para su implementación en la gestión del tráfico urbano, mejorando así la planificación del tráfico, reduciendo tiempos de viaje y emisiones contaminantes, y proporcionando datos confiables para la toma de decisiones en movilidad.

2. ABSTRACT

The acquisition of precise real-time vehicle flow data is essential for the management and optimization of transportation networks. Current methods in Bogotá, such as manual counting and fixed sensors, have significant limitations in coverage and accuracy.

This project addresses these challenges by developing an intelligent real-time vehicle counting system using convolutional neural networks and computer vision. Utilizing the YOLOv8n model, vehicles were detected and classified into three categories: cars, motorcycles, and buses/trucks.

High-resolution videos were captured at a strategic point in Bogotá, and the images were processed to train the model. Data processing techniques included labeling, augmentation, and normalization of images. Evaluation metrics such as precision, recall, mAP, F1 Score, and confusion matrix analysis demonstrated the model's high effectiveness in vehicle detection. Additionally, processing speeds and system efficiency, crucial for real-time applications, were discussed.

The results show a high level of precision and recall, suggesting the system's feasibility for implementation in urban traffic management. This enhances traffic planning, reduces travel times and emissions, and provides reliable data for mobility decision-making.

3. IDENTIFICACIÓN DEL PROBLEMA

La obtención de datos precisos sobre el flujo vehicular en tiempo real resulta indispensable para la gestión y optimización de las redes de transporte, tal como indican estudios de ingeniería de tránsito. Sin embargo, la investigación realizada por medio de artículos y análisis de reportes, permitió determinar que los métodos de aforo actuales en la ciudad presentan limitaciones significativas.

El conteo manual por observación directa tiene problemas de cobertura total de la red vial debido a la cantidad limitada de personal técnico, por otro lado, los sensores fijos instalados únicamente proveen lecturas de volumen de tráfico, más no de composición vehicular. Considerando la naturaleza dinámica del tráfico, se necesitan mediciones casi en tiempo real, o viables con los métodos actuales. Tanto el aforo manual como los sensores fijos presentan restricciones importantes.

Si bien existen desde hace años investigaciones sobre el diseño de algoritmos y programas para el seguimiento vehicular, la mayoría de estos desarrollos no son de código abierto, por lo que los softwares disponibles comercialmente resultan costosos y restrictivos. Esta limitación en el acceso a soluciones tecnológicas constituye un obstáculo para avanzar en sistemas efectivos de medición del tráfico. Este es el problema que se ha identificado y que la presente propuesta busca específicamente solucionar, desarrollando un sistema de aforo vehicular basado en procesamiento de imágenes y redes neuronales, que permite mediciones precisas del tráfico y representa una alternativa accesible frente a las limitaciones actuales.

4. JUSTIFICACIÓN

El crecimiento acelerado de vehículos en las ciudades ha generado múltiples problemas de movilidad, congestión vehicular y contaminación ambiental. Para gestionar de forma eficiente las redes de transporte es fundamental contar con sistemas de monitoreo del tráfico en tiempo real. Sin embargo, los métodos tradicionales de aforo manual y con sensores fijos resultan limitados en términos de precisión, costos y capacidad de obtener mediciones instantáneas.

Ante esta problemática, el desarrollo de sistemas automatizados de aforo vehicular basados en visión artificial y redes neuronales convolucionales representa una solución prometedora. Estas técnicas de inteligencia artificial permiten procesar y analizar imágenes de tráfico para detectar, contabilizar y clasificar vehículos de forma precisa y en tiempo real.

El presente proyecto busca diseñar e implementar un innovador sistema de aforo inteligente mediante el procesamiento de imágenes, que supere las limitaciones de los métodos actuales. Los resultados permitirán mejorar la planificación del tráfico, reducir tiempos de viaje y emisiones contaminantes, así como obtener datos confiables para la toma de decisiones en movilidad.

5. OBJETIVOS

5.1. OBJETIVO GENERAL

Desarrollar un sistema de conteo y clasificación automatizada de vehículos mediante la aplicación de algoritmos de inteligencia artificial y redes neuronales convolucionales sobre secuencias de vídeo de tráfico con el fin de obtener mediciones precisas de flujos vehiculares en tiempo real que permitan mejorar los estudios y modelaciones de ingeniería de tránsito.

5.2. OBJETIVOS ESPECÍFICOS

- Diseñar un modelo de red neuronal convolucional capaz de detectar, contar y clasificar de forma precisa los distintos tipos de vehículos en imágenes de tráfico en tiempo real.
- Optimizar el modelo de red neuronal convolucional para garantizar un rendimiento robusto y confiable en diversas condiciones de iluminación, ángulo de visión y tráfico en tiempo real.
- Evaluar el rendimiento del sistema desarrollado en condiciones reales de Bogotá, utilizando datos de tráfico real.

6. MARCO TEÓRICO

6.1. INTELIGENCIA ARTIFICIAL

"La inteligencia artificial (IA) es una rama de la informática que estudia la creación de sistemas informáticos que exhiben características asociadas con la inteligencia en el comportamiento humano, como el aprendizaje, el razonamiento, la resolución de problemas, la percepción y la capacidad para manipular y mover objetos." (Russell & Norvig, 2020, p. 1) [1]

6.2. VISIÓN POR COMPUTADOR

"La visión por computador (VC) es un campo de la inteligencia artificial que habilita a las computadoras para 'ver', identificar y procesar imágenes de una manera similar a la percepción humana." (Sharma et al., 2020, p. 2) [2]

La visión por computador se aplica en una variedad de áreas, como la ingeniería, el transporte, la seguridad, la robótica y la medicina. En el contexto del transporte, la VC se utiliza para tareas como la detección y conteo de vehículos, el reconocimiento de señales de tráfico, el análisis de flujo vehicular, entre otras.

6.3. APRENDIZAJE PROFUNDO EN LA VISIÓN POR COMPUTADOR

"El aprendizaje profundo es una rama del aprendizaje automático que utiliza redes neuronales artificiales con múltiples capas ocultas para extraer características de

los datos de entrada y realizar tareas como la clasificación, la detección de objetos y el reconocimiento de patrones." (Guo et al., 2016, p. 199) [3]

El aprendizaje profundo ha mejorado significativamente la precisión en tareas de visión por computador, como la detección y el conteo de vehículos. Esto se debe a la capacidad de las redes neuronales para aprender y extraer características complejas de los datos de manera automática, lo que permite una mayor precisión en la identificación de objetos en imágenes y videos.

6.4. FUNCIÓN DE PÉRDIDA

"Una función de pérdida es una medida crítica del error entre las predicciones realizadas por un modelo de aprendizaje automático y los valores reales esperados. Por ejemplo, en el reconocimiento de emociones en el habla, se ha evaluado que las funciones de pérdida basadas en la correlación, como la pérdida de coeficiente de correlación concordante (CCC), ofrecen un mejor rendimiento en comparación con las funciones de pérdida basadas en el error, como el error cuadrático medio (MSE) y el error absoluto medio (MAE)" (Atmaja & Akagi, 2020). [4]

6.5. REDES NEURONALES ARTIFICIALES (RNA)

"Una red neuronal artificial (RNA) es un modelo computacional inspirado en el sistema nervioso biológico, compuesto por un conjunto de unidades de procesamiento simples (neuronas artificiales) que se organizan en capas y están interconectadas por conexiones ponderadas." (Haykin, 2009, p. 28) [5]

Las RNA son ampliamente utilizadas en la visión por computador para tareas como la clasificación de imágenes, la detección de objetos y el reconocimiento de patrones. Algunos ejemplos de aplicaciones de las RNA en la visión por computador incluyen la detección de rostros, el reconocimiento de caracteres ópticos y el análisis de imágenes médicas.

6.6. FEEDFORWARD NEURAL NETWORKS (FNN) RED NEURONAL DE PROPAGACIÓN HACIA ADELANTE

"Una red neuronal feedforward (FNN) es un tipo de red neuronal artificial donde las conexiones entre las unidades no forman ciclos ni bucles. En una FNN, la información se mueve únicamente en una dirección, desde la capa de entrada a través de las capas ocultas (si las hay) hacia la capa de salida." (Schmidhuber, 2015, p. 85) [6]

Las FNN son comúnmente utilizadas en tareas de clasificación y detección, donde se asigna una etiqueta o categoría a una entrada dada. En la visión por computador, las FNN se han aplicado en la clasificación de imágenes, la detección de objetos y el reconocimiento de patrones.

6.7. DEEP BELIEF NETWORKS (DBN) RED DE CREENCIA PROFUNDA

"Una Deep Belief Network (DBN) es un modelo generativo de aprendizaje profundo compuesto por múltiples capas de variables aleatorias binarias con conexiones entre capas sucesivas, pero sin conexiones entre unidades dentro de la misma capa." (Hinton et al., 2006, p. 1) [7]

Las DBN han demostrado ser efectivas en tareas de visión por computador, como el reconocimiento de objetos, la clasificación de imágenes y la detección de rostros. Una de las principales ventajas de las DBN es su capacidad para aprender representaciones jerárquicas de los datos de manera no supervisada, lo que puede ser útil cuando se trabaja con conjuntos de datos grandes y complejos.

6.8. RECURRENT NEURAL NETWORK (RNN) RED NEURONAL RECURRENTE

"Una red neuronal recurrente (RNN) es un tipo de red neuronal artificial donde las conexiones entre las unidades forman un ciclo dirigido, lo que permite que la red tenga una memoria interna y procese secuencias de datos." (Graves, 2012, p. 1) [8]

6.9. CONVOLUTIONAL NEURAL NETWORKS (CNN) RED NEURONAL CONVOLUCIONAL

"Una red neuronal convolucional (CNN) es un tipo de red neuronal artificial diseñada específicamente para procesar datos en forma de matriz, como imágenes." (LeCun et al., 1998, p. 1) [9]

Las CNN están compuestas por diferentes tipos de capas, que incluyen:

a. **Capas Convolucionales**: "Estas capas aplican una operación de convolución a la entrada, lo que permite extraer características de las imágenes de manera eficiente." (Géron, 2019, p. 485) [10]

b. **Capas Agrupadas**: "Las capas de agrupación (pooling) realizan una operación de submuestreo, reduciendo las dimensiones espaciales de los datos." (Géron, 2019, p. 488) [10]

c. **Capas Completamente Conectadas**: "Estas son las capas finales de la CNN, donde las características extraídas por las capas anteriores se combinan para realizar la clasificación." (Géron, 2019, p. 489) [10]

Las CNN han sido ampliamente utilizadas en tareas de visión por computador, como la clasificación de imágenes, la detección de objetos y el reconocimiento de patrones, logrando un alto rendimiento y precisión. [10]

6.10. CONTEO DE MULTITUDES Ó CROWD COUNTING

"El conteo de multitudes (crowd counting) es una tarea de visión por computador que implica contar el número de personas u objetos en una imagen o video. Por ejemplo, el uso de técnicas basadas en redes neuronales convolucionales para generar mapas de densidad ha mostrado ser efectivo para mejorar la precisión del conteo de multitudes en escenas congestionadas" (Zhang et al., 2019). [11]

El conteo de multitudes tiene diversas aplicaciones, como la gestión de la seguridad, el control de flujos peatonales y la planificación urbana. En el contexto del conteo vehicular, técnicas similares se utilizan para contar el número de vehículos en imágenes o videos de tráfico

6.11. ESTIMACIÓN DE LA DENSIDAD

"La estimación de la densidad es un enfoque utilizado en el conteo de objetos en imágenes, donde se estima la densidad de objetos en diferentes regiones de la imagen y luego se integra para obtener un recuento total." (Loy et al., 2013, p. 1)[12]

En el conteo vehicular, la estimación de la densidad se puede aplicar para estimar la cantidad de vehículos en diferentes regiones de una imagen o video de tráfico, lo que permite obtener un recuento total más preciso, especialmente en situaciones de congestión o alta densidad vehicular.

6.12. HAAR CASCADE

"Haar Cascade es un método de detección de objetos basado en características visuales simples denominadas características de Haar, que se utilizan para entrenar un clasificador en cascada." (Viola & Jones, 2001, p. 1) [13]

6.13. ARQUITECTURA DE CLASIFICADOR EN CASCADA

"La arquitectura de clasificador en cascada es un enfoque de aprendizaje automático utilizado para construir un clasificador eficiente, donde se entrena una secuencia de clasificadores débiles y se combinan para formar un clasificador fuerte." (Viola & Jones, 2001, p. 2) [13]

6.14. YOU ONLY LOOK ONCE (YOLO)

"YOLO (You Only Look Once) es un sistema de detección de objetos en tiempo real basado en una red neuronal convolucional, que predice los recuadros delimitadores y las probabilidades de clase de los objetos en una sola evaluación." (Redmon et al., 2016, p. 1) [14]

6.15. BOUNDING BOX (CUADRO DELIMITADOR)

"Un recuadro delimitador (bounding box) es un rectángulo que encierra un objeto de interés en una imagen o video, definido por sus coordenadas (x, y) de la esquina superior izquierda y las coordenadas de la esquina inferior derecha." (Girshick et al., 2014, p. 2) [15]

6.16. ARQUITECTURA DE YOLO

"La arquitectura de YOLO se basa en una red neuronal convolucional que divide la imagen de entrada en una cuadrícula S x S. Cada celda de la cuadrícula predice B recuadros delimitadores, además de las puntuaciones de confianza y las probabilidades de clase." (Redmon et al., 2016, p. 2) [14]

6.17. LOCALIZATION LOSS (PÉRDIDA DE LOCALIZACIÓN)

"La pérdida de localización (localization loss) es una medida del error entre los recuadros delimitadores predichos por el modelo y los recuadros delimitadores reales de los objetos en la imagen." (Redmon et al., 2016, p. 3) [14]

6.18. CONFIDENCE LOSS PÉRDIDA DE CONFIANZA

"La pérdida de confianza (confidence loss) mide la precisión del modelo al predecir la confianza de que un recuadro delimitador contiene un objeto y la precisión de la predicción de la caja que no contiene un objeto." (Redmon et al., 2016, p. 3) [14]

6.19. SINGLE SHOT DETECTOR (SSD)

"El Single Shot Detector (SSD) es un método de detección de objetos en tiempo real que discretiza la salida del espacio de recuadros delimitadores en un conjunto de cajas predefinidas con diferentes relaciones de aspecto y escalas." (Liu et al., 2016, p. 1) [16]

6.20. SEGMENT ANYTHING MODEL (SAM)

"El Segment Anything Model (SAM) es un modelo de aprendizaje profundo desarrollado por Meta AI que permite realizar segmentación de objetos en imágenes de manera eficiente y precisa. Utiliza una arquitectura de transformador

que puede segmentar cualquier objeto en una imagen con solo unos pocos puntos de entrada proporcionados por el usuario." (Cheng et al., 2023, p. 1) [17]

6.21. C2F (CROSS-STAGE PARTIAL BOTTLENECK WITH TWO CONVOLUTIONS)

"El C2f (Cross-Stage Partial Bottleneck with Two Convolutions) es un módulo de construcción utilizado en las redes neuronales convolucionales, específicamente en la arquitectura EfficientNet. Este módulo fue diseñado para mejorar la eficiencia computacional al tiempo que mantiene un alto rendimiento en tareas de visión por computador." (Tan & Le, 2019, p. 4) [18].

6.22. CPU (UNIDAD CENTRAL DE PROCESAMIENTO):

"Una CPU (Unidad Central de Procesamiento) es el componente principal de una computadora que ejecuta instrucciones y realiza cálculos aritméticos y lógicos. Es el cerebro de la computadora y controla todas las operaciones y el flujo de datos." (Patterson & Hennessy, 2019, p. 5) [19].

En el contexto de Google Colab, las instancias de CPU son adecuadas para tareas de procesamiento de datos, entrenamiento de modelos de menor escala y otras operaciones que no requieren un alto rendimiento de cálculo paralelo. Estas instancias de CPU son proporcionadas de forma gratuita por Google Colab.

6.23. GPU (UNIDAD DE PROCESAMIENTO GRÁFICO):

"Una GPU (Unidad de Procesamiento Gráfico) es un procesador especializado diseñado para realizar cálculos intensivos, principalmente enfocados en el procesamiento de gráficos y operaciones paralelas. Las GPU modernas están equipadas con miles de núcleos de procesamiento, lo que las hace ideales para aplicaciones que requieren procesamiento paralelo masivo, como el aprendizaje profundo y el procesamiento de imágenes." (Owens et al., 2008, p. 1) [20].

Google Colab ofrece la posibilidad de habilitar aceleradores de GPU dentro del entorno de Jupyter Notebook. Las GPU son especialmente útiles para entrenar modelos de aprendizaje profundo y redes neuronales, como los utilizados en la detección y conteo vehicular, ya que pueden realizar operaciones matemáticas complejas de forma masivamente paralela, reduciendo significativamente el tiempo de entrenamiento en comparación con las CPU.

7. MATERIALES

7.1. XIAOMI NOTE 8 PRO

El Xiaomi Redmi Note 8 Pro es un smartphone de gama media lanzado en 2019 que ofrece un buen rendimiento y características destacadas. Algunas de sus especificaciones básicas son:

- Pantalla IPS LCD de 6.53 pulgadas con resolución Full HD+
- Procesador Mediatek Helio G90T de ocho núcleos
- 6GB de RAM y 128GB de almacenamiento interno
- Batería de 4,500mAh con carga rápida
- Sistema operativo Android 9 Pie con MIUI 11

El Xiaomi Redmi Note 8 Pro se utilizó principalmente como dispositivo para adquirir datos de vídeo en resolución Full HD 1080p a 60 cuadros por segundo.
La cámara principal de 64 megapíxeles de este smartphone, combinada con su capacidad de grabar video a 1080p/60fps, permitió capturar secuencias de vídeo de alta calidad y con un movimiento fluido, lo cual fue esencial para el análisis y procesamiento de datos requerido en esta investigación

Figura 1

Teléfono Móvil Redmi Note 8 Pro, s. f.



Nota. Por Mercado Libre. (n.d.), 2024[36]. Celular implementado para realizar la captura de videos.

7.2. TRIPODE EN ALUMINIO:

Para la toma de datos de video se utilizó un trípode genérico para smartphones. Este accesorio constaba de:

- Cabezal articulado de 3 vías que permitía ajustar el ángulo de inclinación, rotación y orientación del teléfono.
- Abrazadera o soporte universal con cierre de tornillo donde se montaba firmemente el dispositivo móvil.
- Patas extensibles de aluminio ligero con secciones de altura ajustable para variar la elevación del trípode entre 0.5 y 1.5 metros aproximadamente.
- Cabeza esférica con movimiento de 360 grados para posicionamiento horizontal y vertical preciso.

Figura 2

Trípode



Nota. Por Autor, 2024. Toma de datos con tripode de aluminio y Xiaomi Note 8 pro.

7.3. CABLE USB TIPO C

Una vez obtenidos los vídeos el siguiente paso fue transferirlos a un PC para su posterior procesamiento y análisis.

Para ello, se utilizó un cable USB tipo C para conectar el smartphone al computador. Este cable admite transferencias de datos de alta velocidad gracias a la interfaz USB-C en el Redmi Note 8 Pro.

8. MÉTODOS

8.1. ADQUISICIÓN DE DATOS

La adquisición de datos se llevó a cabo en el puente peatonal ubicado en la avenida Ciudad de Villavicencio con la transversal 70, en el barrio Madelena, de la localidad 19 (Ciudad Bolívar), en la ciudad de Bogotá, Colombia. Este punto fue seleccionado debido a su alto flujo vehicular y la variedad de vehículos que transitan la vía, lo que lo convierte en un lugar idóneo para recopilar información para el estudio.

Figura 3

Ubicación Toma de Datos



Nota. Imagen adaptada de Google maps, 2024 [37]. El Plano adjunto corresponde a la vista en planta del puente peatonal en el cúal se hace la toma de datos y donde se realizará el estudio de movilidad.

Las flechas corresponden al sentido de circulación según la notación del Esquema de movimientos vehiculares RILSA. Donde la flecha azul corresponde al sentido 1 (norte-sur) y la flecha roja corresponde al sentido 2 (sur-norte).

Se realizaron dos tomas de datos distintas, una para aforar el flujo vehicular en sentido norte-sur, y otra en sentido sur-norte. Esta división permitió analizar de manera más precisa los patrones de movimiento en ambas direcciones de tráfico.
El puente peatonal cuenta con un gálibo (altura libre) de 4.50 metros. Para obtener una perspectiva adecuada y capturar la totalidad del flujo vehicular y peatonal, la toma de datos se realizó a una altura aproximada de 1.2 metros sobre el nivel mínimo del puente, lo que proporcionó una altura total de aproximadamente 5.7 metros sobre el nivel de la vía.

Figura 4

Toma de Datos



Nota. Imagen adaptada de Google maps, 2024[37].

La imagen corresponde a la señalización vertical que indica la altura libre (Gálibo) del puente peatonal donde se tomaron los datos, en el sentido 2.

Esta posición elevada fue clave para garantizar una visión completa del área de estudio, evitando obstrucciones visuales y permitiendo una captura óptima de los datos.

La adquisición de los datos se llevó a cabo en las horas de la mañana, específicamente en la franja comprendida entre las 9:00 a.m. y las 11:00 a.m. Este rango horario se eligió debido a que representa uno de los períodos de mayor afluencia de vehículos en la zona, gracias a la proximidad de escuelas, comercios y oficinas.

Durante la toma de datos, las condiciones climáticas fueron soleadas, lo que proporcionó una iluminación adecuada para la captura de imágenes y videos de alta calidad.

Para asegurar la consistencia y precisión en la toma de datos en video, se establecieron referencias visuales y se marcaron referencias físicas in situ. Esto permitió mantener un ángulo de captura similar en todas las sesiones de grabación, lo que facilitó el posterior etiquetado y procesamiento de las imágenes obtenidas.

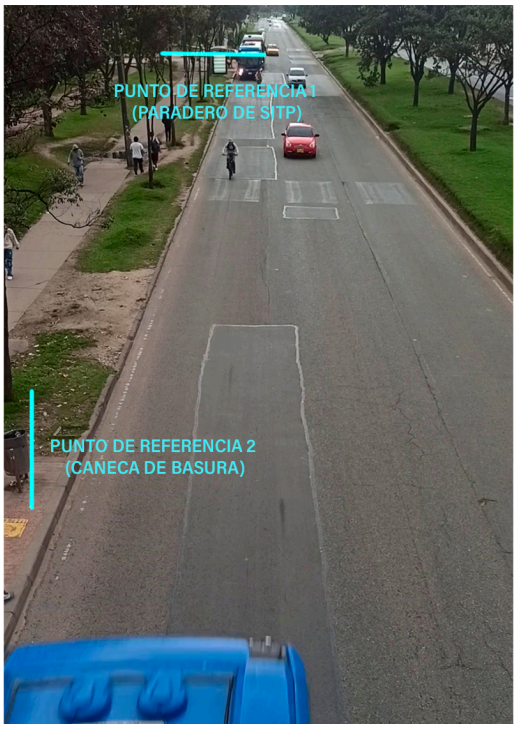
Las referencias visuales consistieron en identificar puntos fijos en el entorno, como señalizaciones o detalles arquitectónicos distintivos. Estos elementos sirvieron como guías para enmarcar adecuadamente el área de interés y mantener una perspectiva coherente en cada toma de datos.

Además, se realizaron marcas físicas en el propio puente peatonal y en sus inmediaciones. Estas marcas, como líneas o puntos de referencia, se colocaron estratégicamente para definir los límites del área de captura y asegurar que el ángulo de la cámara se mantuviera constante en cada sesión.

A continuación, se muestra un ejemplo ilustrativo de las referencias empleadas para la toma de videos:

Figura 5

Puntos de Referencia



Nota. Por autor, 2024. La imagen muestra los puntos de referencia que se tuvieron en cuenta para la toma de video.

En la imagen, se pueden observar las siguientes referencias:

1. Señalizaciones: Señales de tránsito, postes o estructuras fijas que se utilizaron como guías para enmarcar el área de interés.
2. Marcas físicas en el puente: Puntos de referencia dibujados o colocados directamente en el puente peatonal para mantener un ángulo de captura consistente.

Estas referencias visuales y físicas fueron fundamentales para garantizar la coherencia en la recopilación de datos en video. Al mantener un ángulo de captura similar en todas las sesiones, se facilitó el posterior etiquetado y procesamiento de las imágenes obtenidas, lo que contribuyó a la calidad y precisión de los resultados del estudio.

Una vez que se obtienen los minutos de video proyectados, se procede a finalizar la toma de datos en video y desmontar el teléfono móvil del trípode.

8.2. PROCESAMIENTO DE DATOS

Este proceso tiene como objetivo preparar y estandarizar los datos de entrada para mejorar la eficiencia y efectividad del modelo durante el entrenamiento, validación y testeo. En el presente estudio, las imágenes se dividieron en tres categorías principales: entrenamiento (75%), validación (15%) y testeo (10%) para 20.000 imágenes, identificando tres tipos de clases: Carros, Motos y Buses/Camiones.

De tal forma que se pueda mejorar las métricas como la precisión, el recall, la Mean Average Precision (mAP), los Frames Per Second (FPS) y el tiempo de inferencia. Estas métricas evalúan la exactitud con la que el modelo identifica y localiza objetos, así como su velocidad, lo cual es vital para aplicaciones en tiempo real. La precisión y el recall miden la efectividad del modelo al detectar

correctamente los vehículos, mientras que el mAP ofrece una visión comprensiva del rendimiento general a través de varias clases y umbrales de IoU. Por último, los FPS y el tiempo de inferencia son indicadores de la rapidez del modelo, esencial para sistemas de monitoreo vehicular donde la respuesta inmediata es crítica.

8.2.1. Cargar los Datos al Computador

El proceso de Subir los archivos al computador consiste en los siguientes pasos:

- Se conectó un extremo del cable USB tipo C al puerto USB-C del Redmi Note 8 Pro.
- El otro extremo del cable se enchufó a un puerto USB disponible en el PC.
- Una vez establecida la conexión por cable, el computador reconoció al smartphone como un dispositivo de almacenamiento externo.
- Utilizando el administrador de archivos del PC, se accedió al almacenamiento interno del Redmi Note 8 Pro.
- Allí se localizaron los videos grabados en la carpeta DCIM/Camera.
- Finalmente, estos archivos de vídeo en 1080p/60fps se copiaron o transfirieron desde el smartphone al disco duro del PC.

8.2.2. Adobe Express

El proceso para preparar los videoclips en Adobe Express es el siguiente:

Se accedió a la herramienta en línea Adobe Express que se encuentra en este link: (https://www.adobe.com/es/express/feature/video/merge) desde el navegador web del PC. Esta aplicación permite combinar múltiples clips de video en uno solo.

- Carga de los clips: Dentro de Adobe Express, se hizo clic en la opción "Mezclar video" y se seleccionaron todos los clips de video cortos de 5 minutos que se deseaban unir. Los archivos se cargaron en la línea de tiempo del editor.

- Ordenamiento y ajustes: Una vez cargados, los clips se organizaron en el orden correcto en la línea de tiempo. Aquí se pudo visualizar cómo quedaría el video final y realizar ajustes como recortar, cambiar el orden o agregar transiciones entre clips si era necesario.
- Exportación del video final: Finalmente, con los clips ordenados, se procedió a exportar el proyecto como un solo video continuo usando la configuración de resolución 1080p y la tasa de fotogramas de 60fps para mantener la calidad original de las grabaciones.
- De esta manera, utilizando Adobe Express de forma online, fue posible combinar de manera sencilla todos los videos cortos de 5 minutos.

8.2.3. Google Drive

Una vez que se obtiene el videoclip unido, se sube a la plataforma de almacenamiento en la nube google drive. Esto debido a que es un archivo con peso considerable para subirlo directamente al entorno virtual de google collaboratory (esto genera errores y el entorno se desconecta).

8.2.4. Google Collaboratory (Colab)

Google Collaboratory (Colab) es un producto de Google que permite escribir y ejecutar código Python a través de un entorno de Jupyter Notebook alojado en la nube. Es una herramienta popular para el aprendizaje automático, análisis de datos y desarrollo de modelos, ya que ofrece acceso gratuito a recursos de computación en la nube, incluyendo CPU y GPU.

8.2.5. Etiquetado de Imágenes

El etiquetado de imágenes se realiza utilizando la herramienta Grounding Dino (Se ejecuta desde el entorno virtual de google collaboratory) para su posterior verificación en RoboFlow. Este proceso constituye un punto clave para preparar

los datos para el entrenamiento del modelo YOLOv8. Grounding Dino es un modelo de detección de objetos zero-shot que utiliza pares de texto e imagen para generar automáticamente cajas delimitadoras alrededor de los objetos detectados en las imágenes. Al integrar capacidades de comprensión de lenguaje natural, este modelo permite etiquetar objetos sin entrenamiento específico previo para las clases a detectar.

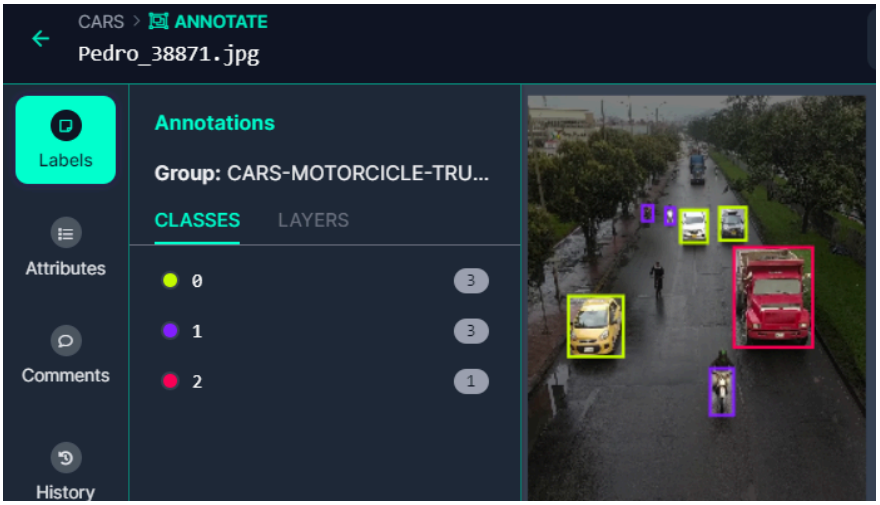
Una vez generadas las cajas delimitadoras con Grounding Dino, el modelo SAM (Segment Anything Model) se utiliza para convertirlas en máscaras de segmentación de instancia, proporcionando una anotación más detallada que simplemente las cajas delimitadoras. Este paso es crucial para obtener una representación precisa de la forma y el tamaño de los objetos dentro de las imágenes. Posteriormente, estas anotaciones se refinan y validan en RoboFlow, donde se ajustan y corrigen para asegurar que los datos estén adecuadamente preparados para entrenar modelos de detección de objetos en tiempo real como YOLOv8. Este proceso de ajuste y validación es esencial para garantizar la calidad y precisión de las etiquetas utilizadas en el entrenamiento del modelo.

8.2.6. Revisión de las Imágenes Etiquetadas por Medio de la Herramienta ROBOFLOW

Roboflow es una plataforma de gestión de datos de visión artificial que facilita el desarrollo y despliegue de modelos de aprendizaje automático, específicamente en el área de la visión por computadora. Se utiliza para la preparación, etiquetado, aumentación y gestión de conjuntos de datos, permitiendo a los desarrolladores entrenar modelos de detección y clasificación de imágenes de manera más eficiente.

Figura 6

Etiquetado de imágenes en la plataforma roboflow



Nota. Por Autor, 2024. Se revisa el etiquetado ya realizado mediante herramienta Roboflow, para después descargar las imágenes y anotaciones optimizadas en un formato creado específicamente para YOLOv8.

8.2.7. Aumento de Imágenes (Data Augmentation)

El aumento de datos es fundamental para mejorar la capacidad del modelo de detectar vehículos en diversas condiciones ambientales y perspectivas. Al aplicar transformaciones como rotaciones, cambios de escala, y ajustes en iluminación, se simulan diferentes condiciones de captura que un vehículo podría enfrentar en el mundo real. Esto es particularmente útil para YOLOv8, ya que el modelo se vuelve más robusto y menos propenso a sobre ajustarse. RoboFlow proporciona herramientas integradas para realizar estas transformaciones de manera eficiente, permitiendo a los usuarios aplicar aumentos con facilidad y precisión.

8.2.8. Redimensionamiento de Imágenes

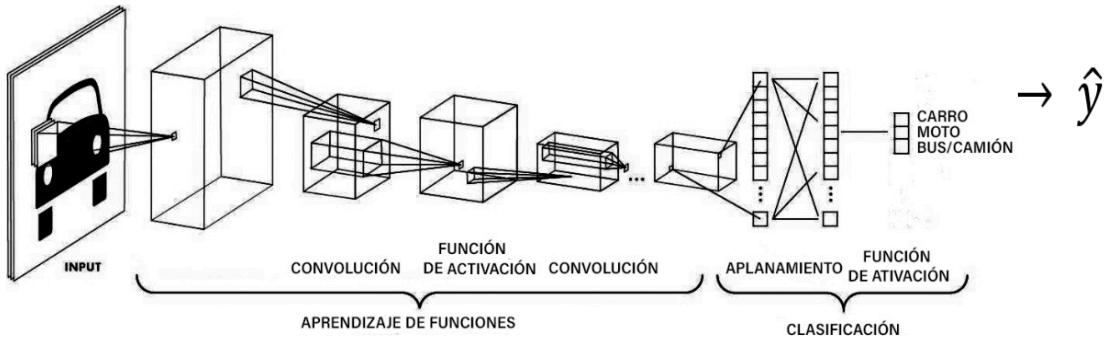
En la aplicación de YOLOv8 para la detección de vehículos, el redimensionamiento de imágenes a un tamaño estándar de 640x640 píxeles es recomendado por el este para maximizar la eficiencia de inferencia. Este preprocesamiento se logra mediante la herramienta de Roboflow “redimensionar imágenes”, que trata los datos para que sean uniformes. Este proceso ayuda a la interpolación bilineal, la cual equilibra la calidad visual y el rendimiento computacional, manteniendo la integridad necesaria para la detección efectiva de características específicas.

8.3. ENTRENAMIENTO DEL MODELO

El entrenamiento de la red neuronal convolucional se centra en la clasificación y detección de vehículos a partir de imágenes de 640x640 píxeles. YOLOv8 utiliza una arquitectura compuesta por múltiples capas de convolución que ajustan automáticamente los valores de los núcleos (kernels) para identificar características clave de los vehículos, como la forma y el tamaño. Estas capas, junto con las de agrupación, procesan las imágenes reduciendo progresivamente su tamaño pero conservando los detalles esenciales para la detección efectiva. A medida que se añaden más capas, la red se profundiza, permitiendo una representación más detallada y facilitando la identificación precisa de diversos tipos de vehículos en diferentes condiciones ambientales. El entrenamiento se realiza en plataformas que soportan alta carga computacional, como GPUs disponibles en Google Colab, optimizando tanto la precisión de la detección como la velocidad, crucial para aplicaciones en tiempo real (Ultralytics, 2023)[21]. El código implementado se muestra en el apéndice A.

Figura 7

Aprendizaje de red neuronal



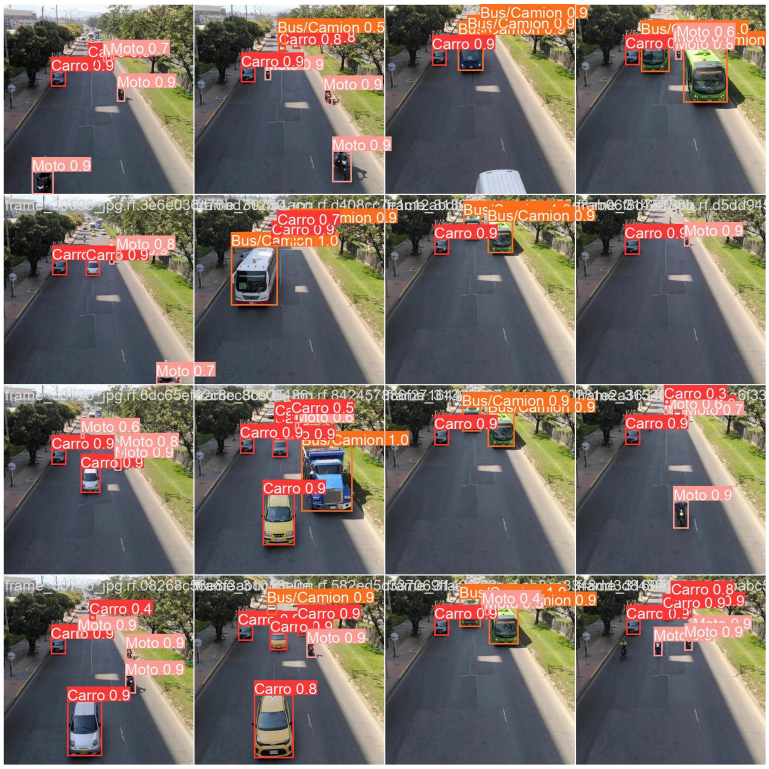
Nota. Imagen adaptada de MathWorks, 2017[38]. Se muestra cómo la neurona aprende e interpreta la información que se le suministra.

8.3.1. Imágenes de Entrenamiento, Validación y Testeo

Los conjuntos de entrenamiento, validación y testeo son fundamentales para desarrollar un modelo robusto y fiable. El conjunto de entrenamiento se utiliza para ajustar los pesos del modelo, permitiéndole aprender a partir de los datos proporcionados. El conjunto de validación es crucial para ajustar hiperparámetros y evaluar la capacidad de generalización del modelo sin comprometer la integridad del conjunto de testeo. Este último se utiliza para evaluar el rendimiento final del modelo y asegurar que funcione correctamente bajo condiciones similares a las que encontrará en producción. Es esencial que estos conjuntos sean representativos de la variedad real de condiciones que los vehículos experimentarán, garantizando así que los ajustes en el modelo conduzcan a mejoras en la generalización y no solo en el rendimiento en los datos conocidos. Este enfoque de evaluación rigurosa es fundamental para desarrollar un sistema de detección de vehículos preciso y fiable, capaz de operar eficazmente en un amplio rango de entornos.

Figura 8

Imágenes de validación



Nota. Por Autor, 2024. Imágenes usadas en el entrenamiento del modelo.

8.3.2. Parámetros del Modelo

Los parámetros del modelo en YOLOv8, que incluyen los pesos y segmentos de las neuronas, son fundamentales para la precisión del modelo. Durante el entrenamiento, estos parámetros se ajustan continuamente mediante algoritmos de optimización como SGD (Stochastic Gradient Descent) o Adam (Adaptive Moment Estimation) para minimizar la función de pérdida, que mide el error entre las predicciones del modelo y las verdaderas anotaciones de los objetos en las imágenes. El número y la complejidad de estos parámetros dependen de la profundidad y el diseño de la red, y su optimización efectiva es clave para lograr

un alto rendimiento en la detección precisa de vehículos (Redmon & Farhadi, 2018)[22].

8.3.3. Número de Épocas (Epochs)

El número de épocas es un factor crucial para determinar cuántas veces el modelo observa el conjunto completo de datos, lo que influye directamente en su capacidad de aprendizaje y generalización. Cada época representa una iteración completa sobre el conjunto de datos, donde el modelo ajusta sus pesos basándose en el error entre las predicciones y las etiquetas reales. Determinar el número óptimo de épocas es esencial: un número insuficiente puede resultar en un aprendizaje inadecuado, mientras que un exceso puede causar sobreajuste, donde el modelo aprende el ruido y los detalles específicos del conjunto de entrenamiento, en detrimento de la generalización. Por lo tanto, es crucial encontrar un valor óptimo para las épocas, generalmente recomendado entre 60 y 90 épocas (Ultralytics, 2023)[21]. Este equilibrio asegura que el modelo esté adecuadamente ajustado para identificar vehículos de manera precisa en diversas condiciones, sin comprometer su efectividad en datos no vistos previamente.

8.3.4. Tamaño de Lote (Batch Size)

Un parámetro importante a tener en cuenta es el tamaño del lote, o "batch size", ya que impacta directamente la eficiencia del entrenamiento, la estabilidad de la convergencia y la calidad del modelo final. Un lote más grande permite al modelo estimar el gradiente de la función de pérdida con mayor precisión, lo que puede conducir a un entrenamiento más estable y a una convergencia más rápida hacia un mínimo de pérdida. Sin embargo, los lotes grandes también incrementan la carga computacional y la demanda de memoria, lo que puede ser problemático en dispositivos con recursos limitados. Por otro lado, un tamaño de lote más pequeño

puede hacer que el entrenamiento sea más ruidoso o inestable, ya que cada estimación del gradiente se basa en menos datos, aunque esto puede ser beneficioso para escapar de mínimos locales en la superficie de error. Además, ajustar el tamaño del lote adecuadamente puede ayudar a prevenir el sobreajuste, ya que lotes más pequeños proporcionan un ruido inherente durante el entrenamiento que actúa como un regularizador. En la práctica, la elección del tamaño del lote debe equilibrar los recursos disponibles, la necesidad de precisión y el problema a tratar (Goodfellow, Bengio & Courville, 2016)[23].

8.3.5. Elección del Modelo

En la librería de Ultralytics se presenta una tabla detallada que compara diversos modelos de la familia YOLOv8, diseñados para la detección de objetos en tiempo real. Estos modelos, identificados como YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l y YOLOv8x, ofrecen una gama de propiedades que permiten a los investigadores y desarrolladores seleccionar el más adecuado según sus necesidades específicas. La tabla, mostrada en la figura 9, resume las características clave de cada modelo, incluyendo el tamaño de las imágenes de entrada, la precisión promedio (mAP), la velocidad de inferencia tanto en CPU como en GPU, el número de parámetros y las operaciones de punto flotante por segundo (FLOPs).

Cada modelo de YOLOv8 está optimizado para un equilibrio particular entre precisión y eficiencia. Por ejemplo, el modelo YOLOv8n (nano) es el más ligero y rápido, ideal para aplicaciones en dispositivos con recursos limitados. En contraste, el modelo YOLOv8x (extra-large) ofrece la mayor precisión, aunque a costa de mayores requerimientos computacionales y tiempos de procesamiento más largos. Esta variedad permite adaptarse a diferentes escenarios, desde la detección vehicular en tiempo real hasta aplicaciones más complejas que requieren una alta precisión.

Figura 9

Diferentes modelos de YOLOv8

Model	SIZE (PIXELS)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.2	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Nota. Por Ultralytics, 2023[21]. Muestra las diferentes características que posee cada modelo de YOLOv8.

Model: Identifica la versión específica de YOLOv8. Las versiones van desde 'n' (nano) hasta 'x' (extra-large), representando un aumento en la complejidad y capacidad del modelo.

Size (pixels): El tamaño de las imágenes de entrada que el modelo utiliza. En este caso, todas las versiones utilizan imágenes de 640 x 640 píxeles.

mAPval 50-95: El "mean Average Precision" (mAP) en un rango de IoU (Intersection over Union) de 50% a 95%. Es una métrica estándar para evaluar la precisión de los modelos de detección de objetos. Un mAP más alto indica una mayor precisión.

Speed CPU ONNX (ms): El tiempo de inferencia en milisegundos cuando el modelo se ejecuta en una CPU utilizando el formato ONNX (Open Neural Network Exchange). Un tiempo menor indica un modelo más rápido.

Speed A100 TensorRT (ms): El tiempo de inferencia en milisegundos cuando el modelo se ejecuta en una GPU Nvidia A100 utilizando TensorRT. Un menor tiempo de inferencia indica un rendimiento más rápido en hardware de alto rendimiento.

Params (M): El número de parámetros del modelo en millones. Más parámetros generalmente significan un modelo más complejo y capaz, pero también más grande y más lento.
FLOPs (B): El número de operaciones de punto flotante por segundo (FLOPs) en miles de millones. Una mayor cantidad de FLOPs indica un modelo que requiere más operaciones de cálculo.

8.3.5.1. YOLOv8n

El modelo YOLOv8n (nano) es el más liviano y rápido. Tiene el menor número de parámetros y FLOPs, lo que lo hace adecuado para dispositivos con recursos limitados y aplicaciones que requieren alta velocidad de procesamiento. Sin embargo, su precisión (mAP) es la más baja entre los modelos comparados.

8.3.5.2. YOLOv8s

El modelo YOLOv8s (small) ofrece un equilibrio entre velocidad y precisión. Tiene una mayor precisión en comparación con el modelo nano, pero también requiere más parámetros y FLOPs.

8.3.5.3. YOLOv8m

El modelo YOLOv8m (medium) mejora aún más la precisión y tiene un número moderado de parámetros y FLOPs. Este modelo es adecuado para aplicaciones que pueden equilibrar la velocidad y la precisión sin una limitación estricta de recursos

8.3.5.4. YOLOv8l

El modelo YOLOv8l (large) proporciona una alta precisión a costa de una mayor complejidad y tiempo de procesamiento. Es adecuado para aplicaciones donde la precisión es crítica y los recursos de hardware no son una limitación.

8.3.5.5. YOLOv8x

El modelo YOLOv8x (extra-large) es el más preciso, pero también el más grande y lento. Es ideal para entornos donde la máxima precisión es esencial y se dispone de recursos computacionales avanzados.

La importancia de estas características se refleja en su impacto directo sobre la eficiencia y la precisión del aprendizaje y la inferencia en tareas de visión por computadora. La velocidad de inferencia es crucial para aplicaciones en tiempo real, como la monitorización del tráfico, donde cada milisegundo cuenta. Del mismo modo, la cantidad de parámetros y FLOPs influye en la capacidad del modelo para aprender y generalizar a partir de los datos de entrenamiento, así como en su capacidad para manejar grandes volúmenes de datos en escenarios prácticos.

Por lo tanto, la elección del modelo adecuado de YOLOv8 no solo depende de los recursos disponibles, sino también de los requisitos específicos del proyecto en términos de precisión y rendimiento. La tabla proporcionada por Ultralytics sirve como una guía esencial para tomar decisiones informadas, asegurando que el modelo seleccionado cumpla con los objetivos del proyecto de manera eficiente y efectiva.

Para el modelo realizado, se optó por elegir el YOLOv8n debido a sus ventajas para la detección y conteo de vehículos en tiempo real, su alta velocidad de procesamiento, con un tiempo de inferencia de solo 0.99 ms en una GPU A100, lo

convierte en el más rápido de todos los modelos YOLOv8. Esta característica es crucial para aplicaciones en tiempo real como el conteo vehicular, donde cada milisegundo cuenta. Además, este destaca por su baja complejidad y requerimientos computacionales. Con tan solo 3.2 millones de parámetros y 8.7 billones de operaciones de punto flotante por segundo (FLOPs), este modelo es altamente eficiente, lo que permite su implementación en dispositivos con recursos limitados.

Por lo cual, a pesar de ser el modelo más ligero de la familia, el YOLOv8n ofrece una precisión suficiente para la tarea de detección de vehículos, con un mAP (mean Average Precision) de 37.3%. Si bien esta precisión es menor que la de los modelos más complejos, sigue siendo adecuada para la detección de vehículos en diversas condiciones, lo que cumple con los requisitos de nuestro proyecto.

8.3.6. Redimensionamiento

Una vez iniciado el entrenamiento teniendo claro los diferentes aspectos que pueden influir en un buen modelamiento. Las imágenes ingresadas son transformadas utilizando convoluciones para extraer y resaltar patrones locales binarios, lo cual es esencial para el procesamiento y clasificación precisos en entornos dinámicos y variados. El uso de esta técnica de interpolación asegura que las características esenciales sean destacadas adecuadamente a través de las transformaciones convolucionales, mejorando así la precisión de la clasificación vehicular (Ultralytics, 2023)[21].

8.3.7. Normalización

En el desarrollo de modelos de detección de vehículos como YOLOv8, la normalización de imágenes es crucial para asegurar que los datos de entrada sean consistentes y manejables para la red neuronal. Este proceso involucra

escalar los valores de los píxeles de cada imagen al rango [0,1], una práctica estándar en el procesamiento de imágenes para aprendizaje profundo. Dicha normalización se realiza dividiendo cada valor de píxel por 255, dado que los píxeles de una imagen típicamente varían entre 0 a 255.

Este ajuste es vital porque las funciones de activación, como Softmax en la capa de salida de YOLOv8, operan más eficientemente cuando los valores de entrada están dentro de este rango limitado. Softmax convierte los logits, que son las entradas numéricas a la función, en una distribución probabilística donde cada valor representa la probabilidad de que la entrada pertenezca a una clase específica. Al normalizar las imágenes, se reduce el riesgo de que ciertas características dominen el aprendizaje debido a diferencias de escala, permitiendo que el modelo se enfoque más en aprender las características útiles de las imágenes en lugar de adaptarse a rangos de valores de píxeles variables.

Además, la normalización ayuda a mejorar la convergencia durante el entrenamiento del modelo, lo que resulta en un aprendizaje más rápido y estable. Esto es particularmente importante en aplicaciones de detección en tiempo real, donde el modelo necesita operar de manera eficiente y con alta precisión bajo diferentes condiciones de iluminación y entornos (Ultralytics, 2023)[21].

8.3.8. Entorno de Ejecución

Google Colab ofrece un entorno de ejecución robusto y accesible para entrenar modelos de aprendizaje profundo como YOLOv8, especialmente en aplicaciones como la clasificación de vehículos. Proporciona acceso a recursos computacionales avanzados como GPU y TPU, lo cual es esencial para manejar operaciones intensivas y grandes volúmenes de datos. Esto es muy beneficioso debido a su capacidad para realizar operaciones matriciales y vectoriales a gran escala de manera eficiente, lo que puede acelerar significativamente los tiempos

de entrenamiento comparado con el uso tradicional de CPU. Colab integra estos recursos directamente en su entorno de cuadernos Jupyter, lo que permite centrarse en la experimentación y optimización de modelos sin preocuparse por la infraestructura subyacente (Google. (n.d.). Colaboratory, 2024)[24].

Figura 10

GPU NVIDIA T4



Nota. Por advantech, 2024[39]. GPU prestado por Colab, con 16GB de memoria.

8.4. ARQUITECTURA DE LA RED NEURONAL CONVOLUCIONAL (RNC)

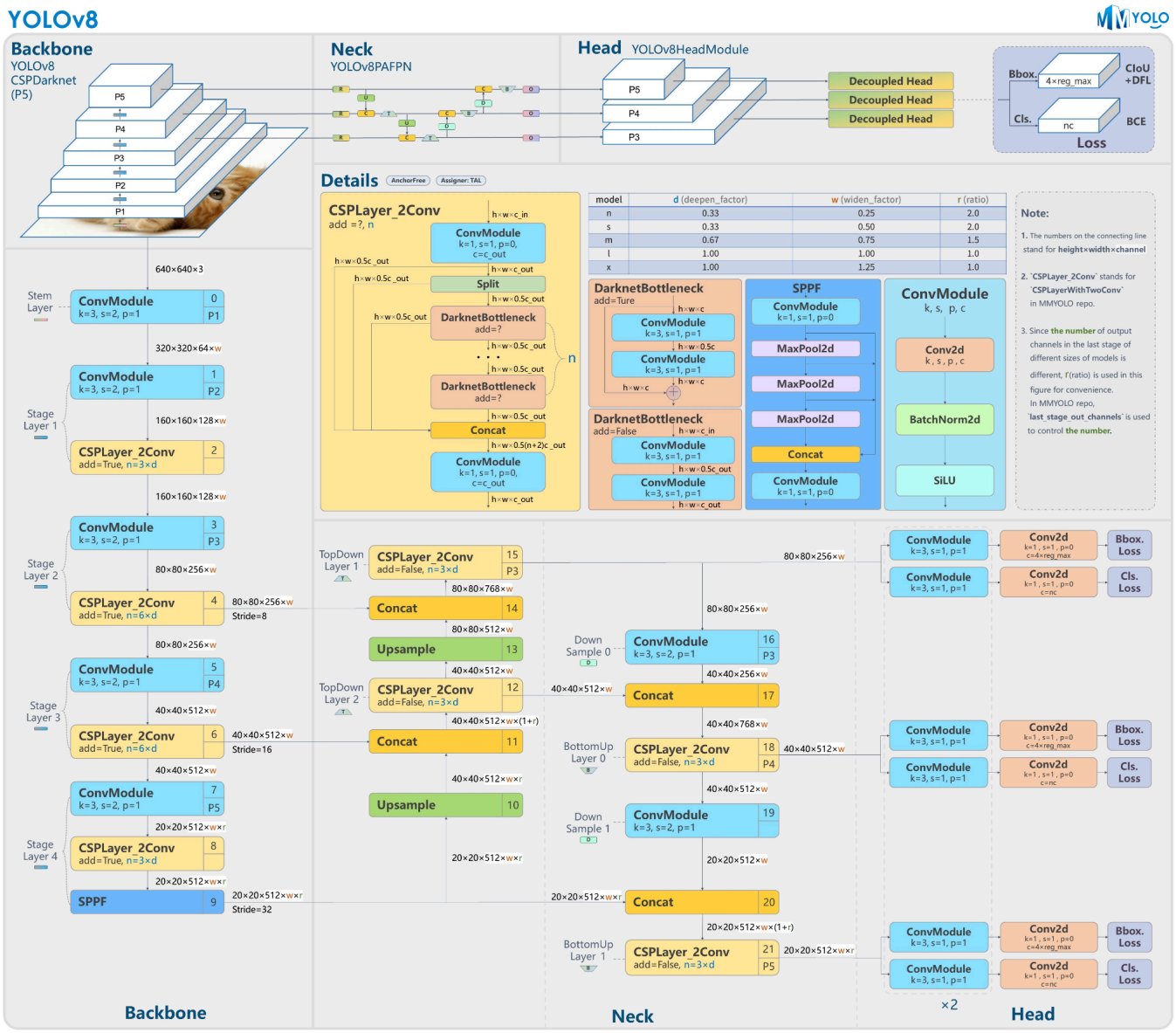
La arquitectura de la red neuronal convolucional (RNC) en YOLOv8 es una evolución significativa de las versiones anteriores de YOLO, diseñada específicamente para mejorar la eficiencia y precisión en tareas de detección en tiempo real. YOLOv8 emplea una arquitectura de red profunda que incluye múltiples capas convolucionales, capas de pooling y capas completamente conectadas, organizadas de manera estratégica para maximizar el rendimiento del modelo.

YOLOv8 introduce el módulo C2f (cross-stage partial bottleneck with two convolutions), que combina características de alto nivel con información contextual para mejorar la precisión de detección. Esta arquitectura comprende alrededor de 75 capas convolucionales, cada una aplicando varios filtros a la imagen de entrada para extraer características de alto nivel como bordes, texturas y formas. Estas características son luego agrupadas por las capas de pooling para reducir la dimensionalidad y mejorar la invarianza a la escala y la posición. Las capas completamente conectadas al final de la red ayudan en la clasificación final de los objetos detectados (Juan Terven, 2023)[25].

La combinación extensa de estas capas permite que YOLOv8 detecte objetos con diferentes escalas y en diversas posiciones con alta precisión. Esta arquitectura es crucial para aplicaciones como el aforo vehicular, ya que asegura que el modelo pueda operar eficientemente incluso en entornos dinámicos y variados.

Figura 11

Arquitectura detallada de YOLOv8.



Nota. Por by MMYOLO, 2023[41]. Se proporciona una visión completa de la arquitectura de YOLOv8, dividida en tres partes principales: Backbone, Neck, y Head.

8.4.1. Backbone (YOLOv8 CSPDarknet)

- Stem Layer: La entrada de la red es una imagen de 640x640 que pasa por la primera capa convolucional.
- Stage Layers (1-4): Cada etapa consiste en varias capas convolucionales (ConvModule) y capas de agrupamiento de pasos cruzados (CSP). Las capas CSP dividen y fusionan características para mejorar la eficiencia y la capacidad de generalización del modelo.

Stage 1: Procesa imágenes de 320x320 con 128 filtros.

Stage 2: Reduce la dimensión espacial y aumenta la profundidad.

Stage 3: Procesa imágenes de 160x160 con 256 filtros.

Stage 4: Finaliza con imágenes de 80x80 con 512 filtros y aplica un módulo SPPF (Spatial Pyramid Pooling Fast) para mejorar la invarianza a la escala.

8.4.2. Neck (YOLOv8PAFPN)

El cuello conecta el Backbone con la cabeza y realiza la fusión de características a diferentes escalas:

TopDown Layers: Utiliza operaciones de concatenación y muestreo para combinar características de diferentes niveles de la red.

BottomUp Layers: Ayuda a refinar las características mediante la combinación de información de diferentes resoluciones.

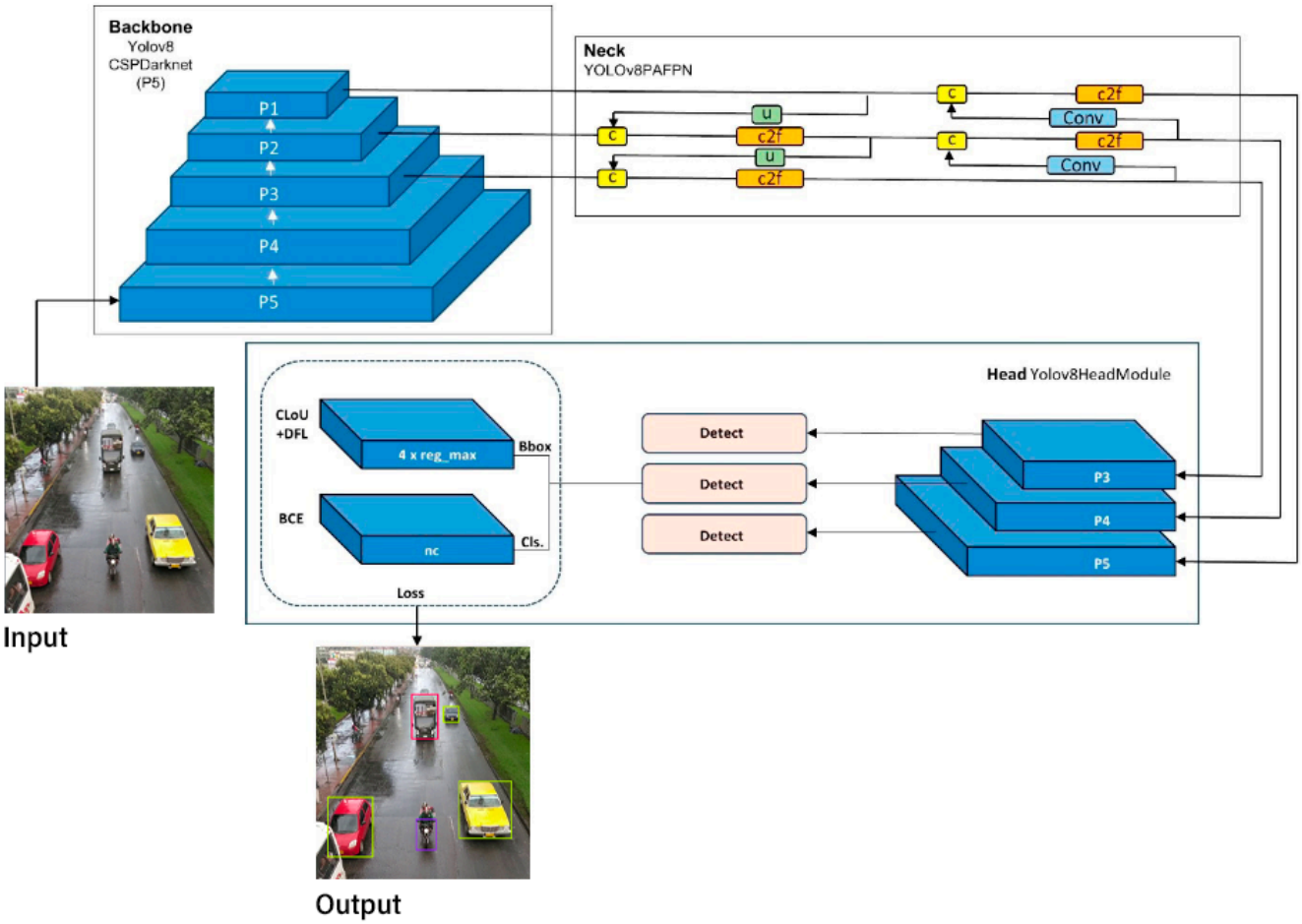
Concat y Upsample: Estas operaciones aseguran que las características se fusionen adecuadamente para capturar tanto detalles finos como contextuales.

8.4.3. Head (YOLOv8HeadModule)

- Detection Heads: Para cada nivel de resolución (P3, P4, P5), se aplican capas convolucionales y de ajuste para producir las predicciones finales. Estas incluyen la caja delimitadora (bbox), la clasificación de objetos (cls), y la pérdida de regresión (loss).
- Conv2d Layers: Aplican filtros para extraer las características finales.
- Bbox Loss, Cls Loss: Se calculan las pérdidas de las cajas delimitadoras y la clasificación, respectivamente, para ajustar el modelo durante el entrenamiento.
- Decoupled Heads: Separan las tareas de clasificación y regresión para mejorar la precisión.

Figura 12

Arquitectura de YOLOv8.



Nota. Imagen adaptada de Brahm Davea, 2023[40]. Arquitectura para modelo de detección vehicular.

8.4.4. Función de Activación

En YOLOv8, las funciones de activación desempeñan un papel crucial al introducir no linealidades necesarias para el aprendizaje de patrones complejos en los datos de entrada, lo cual es esencial para la eficacia del modelo en tareas de detección de objetos como la clasificación de vehículos. Una de las funciones de activación

más utilizadas en YOLOv8 es la ReLU (Rectified Linear Unit). Esta función se define matemáticamente como f (x) = máximo (0 ,X), lo que significa que proporciona la salida directa para todos los valores de entrada positivos y cero para los negativos.

La popularidad de la función ReLU en la arquitectura de YOLOv8 se debe a su capacidad para acelerar la convergencia durante el entrenamiento y mitigar el problema del gradiente desvanecido, un desafío común en redes neuronales profundas donde los gradientes pueden volverse muy pequeños, efectivamente deteniendo el modelo. durante el entrenamiento. Además, ReLU ayuda a mantener la eficiencia computacional, lo que es crucial en aplicaciones en tiempo real como la detección y clasificación de vehículos. La capacidad de ReLU para tratar de manera efectiva variaciones de iluminación y otros cambios visuales complejos en las imágenes es especialmente valiosa, ya que asegura que el modelo no solo aprende bien los datos durante el entrenamiento, sino que también generalice eficazmente a nuevas imágenes en entornos. de aplicación práctica (LeCun, Bengio, & Hinton, 2015)[26].

8.4.5. Compilación del Modelo

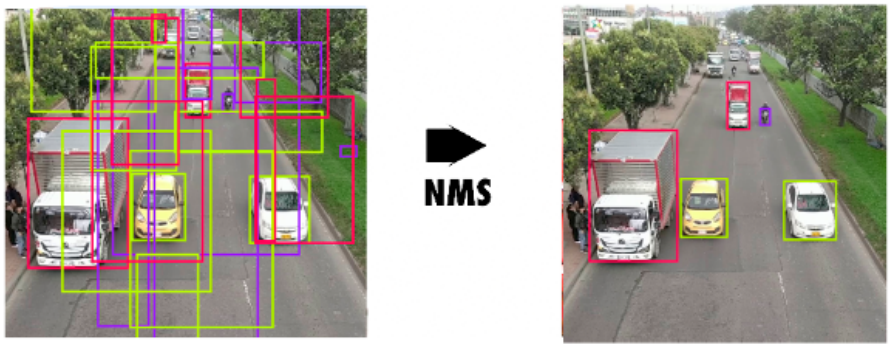
La compilación del modelo en YOLOv8 es un proceso integral que prepara el modelo para el entrenamiento y la inferencia efectivas. Este proceso implica la selección cuidadosa de la función de pérdida y el algoritmo de optimización, elementos clave que dictan cómo el modelo ajusta sus parámetros internos y responde al error calculado. Comúnmente, se utiliza la pérdida de entropía cruzada para la clasificación y detección de objetos, ya que evalúa eficazmente la discrepancia entre las probabilidades predichas y las etiquetas reales. En cuanto a los algoritmos de optimización, opciones como SGD o Adam son elegidas en

función de la eficiencia computacional deseada y la velocidad de convergencia, adecuándose a la estructura y el tamaño del modelo.

Además, la compilación de YOLOv8 incluye la configuración de parámetros específicos para la detección, como los umbrales de confianza para las predicciones y la aplicación de técnicas como la supresión de no máximos (NMS). NMS es vital para eliminar detecciones redundantes y asegurar que cada objeto detectado sea representado por la caja delimitadora más precisa, lo cual es fundamental para mantener la precisión y reducir los falsos positivos en aplicaciones de monitoreo de tráfico (Bochkovskiy, Wang, & Liao, 2020)[27].

Figura 13

NMS (Supresión de no máximos)



Nota. Por Autor, 2024. Supresión no máxima (NMS). a) Muestra el resultado típico de un modelo de detección de objetos que contiene múltiples cuadros superpuestos. b) Muestra la salida después de NMS.

8.5. ESTADÍSTICOS DE CONTROL

En la evaluación del modelo YOLOv8 para la detección y conteo vehicular, se utilizan varias métricas y estadísticos de control que proporcionan una visión

integral de su rendimiento y áreas de mejora. A continuación, se describen en detalle estas métricas, su impacto en el modelo, cómo mejorarlas y cómo interpretar los resultados:

8.5.1. Precisión(P) y Recall(R)

Precisión (P): Indica la proporción de verdaderos positivos entre todas las detecciones positivas, reflejando la exactitud del modelo. Un valor alto de precisión indica que cuando el modelo predice la presencia de un objeto, es muy probable que esa predicción sea correcta.

$$P = \frac{TP}{TP+FP}$$

Donde TP es el número de verdaderos positivos (Son las instancias en las que el modelo predice correctamente la presencia de un objeto que realmente está presente en la imagen.) y FP es el número de falsos positivos (Son las instancias en las que el modelo predice la presencia de un objeto que en realidad no está presente en la imagen.), (Saito & Rehmsmeier, 2015)[28].

Recall (R): Mide la proporción de verdaderos positivos entre el total de instancias que realmente son positivas. Indica la capacidad del modelo para encontrar todas las instancias positivas. Un valor alto de recall indica que el modelo es capaz de detectar la mayoría de los objetos presentes en la imagen (Wang, X., Marko, R., Liu, Z., & Lin, K. 2021)[29].

$$R = \frac{TP}{TP+FN}$$

Donde FN es el número de falsos negativos (Son las instancias en las que el modelo no detecta un objeto (vehículo) que realmente está presente en la imagen.)

8.5.2. Mean Average Precision (mAP)

mAP@0.5: Es una métrica que mide la precisión promedio de un modelo de detección de objetos a un umbral de IoU (Intersección sobre Unión) de 0.5. Esta métrica evalúa la capacidad del modelo para localizar y clasificar correctamente los objetos en una imagen. La fórmula para IoU es:

$$IoU = \frac{\text{Área de la intersección}}{\text{Área de la unión}}$$

Para calcular mAP@0.5, se calcula la precisión promedio (AP) para cada clase a un IoU de

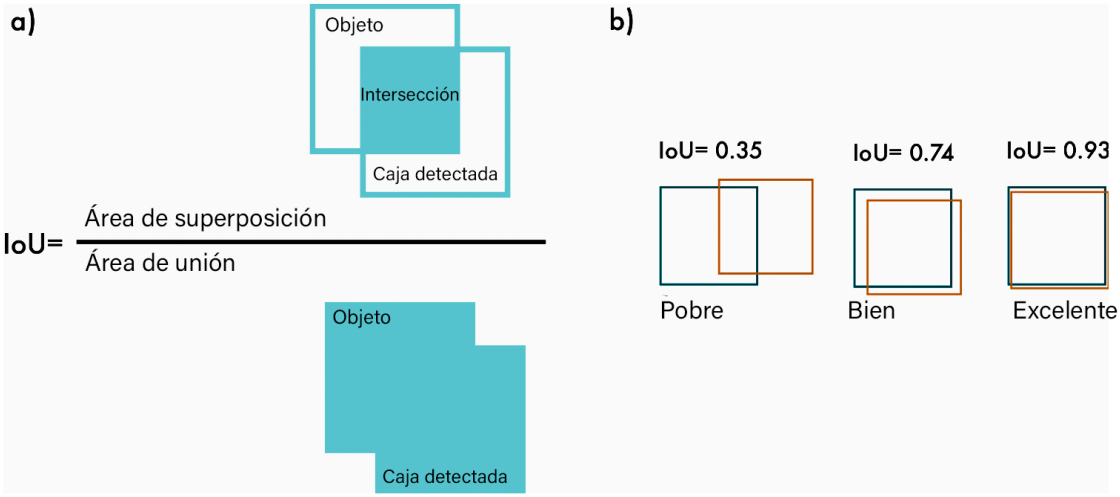
0.5 y luego se promedian estas precisiones. La fórmula para mAP@0.5 es:

$$mAP@0.5 = \frac{1}{n} * \sum_{i=1}^{n} AP_i$$

donde n es el número de clases y APi es la precisión promedio para la clase i. Un valor alto de mAP@0.5 indica que el modelo tiene un buen desempeño en la detección y clasificación de objetos a un nivel de superposición razonable. Los valores recomendados para mAP@0.5 suelen estar por encima del 70% (0.7) para considerarse buenos en aplicaciones prácticas, y valores por encima del 90% (0.9) se consideran excelentes (Juan Terven, 2023)[25].

Figura 14

Precisión Promedio(AP) e Intersección sobre Unión (IoU)



Nota. Imagen adaptada de Juan Terven, 2023[25]. (a) El IoU se calcula dividiendo la intersección de las dos casillas por la unión de las casillas; (b) ejemplos de tres valores de IoU diferentes para diferentes ubicaciones de cajas.

mAP@0.5:0.95: Es una métrica más rigurosa que mide la precisión promedio en múltiples umbrales de IoU, desde 0.5 hasta 0.95 en incrementos de 0.05. Esta métrica proporciona una evaluación más completa del rendimiento del modelo al considerar diferentes niveles de superposición entre las cajas delimitadoras predichas y las reales. Los valores recomendados para mAP@0.5:0.95 suelen ser más bajos que mAP@0.5 debido a la mayor exigencia de la métrica, con valores por encima del 50% (0.5) considerados buenos y valores por encima del 70% (0.7) considerados excelentes (Huang, J. 2017)[30].

8.5.3. F1 Score

El F1 Score varía entre 0 y 1, donde 1 representa el mejor rendimiento posible y 0 el peor. Un F1 Score alto indica que el modelo tiene un buen equilibrio entre

precisión y recall, lo cual es crucial en aplicaciones prácticas donde ambos aspectos son importantes. En general, un F1 Score superior a 0.7 se considera bueno y valores superiores a 0.9 se consideran excelentes en la mayoría de los casos de uso (Saito, T., & Rehmsmeier, M. 2015)[28].

$$F1 = 2 * \frac{P*R}{P+R}$$

8.5.4. Confianza (Confidence)

En un modelo de detección de objetos, como YOLOv8, se refiere al score de confianza asignado a cada predicción, indicando la certeza del modelo sobre la presencia de un objeto en una determinada ubicación. Un valor alto de confianza sugiere que el modelo está muy seguro de que la predicción es correcta, lo que generalmente se traduce en una menor tasa de falsos positivos.

Un umbral de confianza más alto puede aumentar la precisión al reducir las detecciones incorrectas, pero también puede disminuir el recall al no detectar algunos objetos reales. Inversamente, un umbral más bajo puede aumentar el recall al detectar más objetos, pero también puede incrementar la cantidad de falsos positivos, reduciendo así la precisión.

Un enfoque común es el análisis de la curva precisión-recall para determinar el punto de operación óptimo que maximiza el rendimiento del modelo según las métricas de interés (Redmon & Farhadi, 2018)[31].

8.5.5. Matriz de Confusión

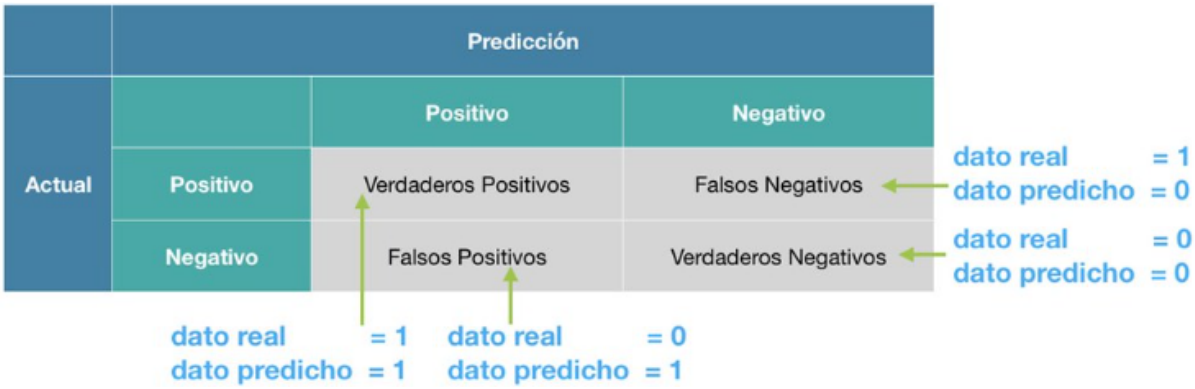
Es una herramienta crucial en la evaluación de modelos de detección de objetos, proporcionando una representación visual de las predicciones correctas e incorrectas del modelo. Esta matriz se estructura en una tabla donde las filas representan las instancias reales y las columnas representan las predicciones del modelo. Los elementos de la matriz indican el número de veces que cada

combinación de predicciones y resultados reales ocurre, lo que facilita la identificación de patrones de error entre diferentes clases. En una matriz de confusión ideal, la diagonal principal (que representa los verdaderos positivos) contendría los valores más altos, mientras que los elementos fuera de la diagonal (que representan falsos positivos, falsos negativos y verdaderos negativos) serían cercanos a cero.

Los valores recomendados en una matriz de confusión dependen del contexto y del equilibrio deseado entre precisión y recall. En general, se busca maximizar los verdaderos positivos y minimizar los falsos positivos y falsos negativos. La interpretación de la matriz de confusión incluye el cálculo de métricas derivadas como la precisión (P), el recall (R), y el F1 Score. Por ejemplo, en la detección y conteo vehicular, un alto número de verdaderos positivos para cada clase de vehículo (carro, moto, bus/camión) indica que el modelo está realizando bien las detecciones, mientras que un alto número de falsos positivos o falsos negativos puede señalar problemas específicos que requieren ajustes en el modelo o en los datos de entrenamiento (Kohavi & Provost, 1998)[32].

Figura 15

Matriz de Confusión



Nota. González, L. 2019[33]. Se define 1 como verdadero y 0 como falso. Por ejemplo, en la celda de verdaderos positivos, se tiene el caso donde el dato es realmente positivo y el modelo también lo ha identificado correctamente como positivo, por lo cual entre mayor sea este valor mayor será su precisión.

8.5.6. Losses: Box Loss, Cls Loss, Dfl Loss

Las métricas de pérdida (losses) son cruciales para evaluar y mejorar el rendimiento de los modelos de detección de objetos como YOLOv8, ya que cada tipo de pérdida aborda diferentes aspectos del modelo.

- Box Loss: Mide el error en la predicción de las coordenadas de las cajas delimitadoras, influenciando directamente la precisión de localización de los objetos detectados. Esta métrica evalúa la discrepancia entre las cajas predichas y las cajas reales en términos de su posición y tamaño. Un valor bajo de Box Loss indica que las predicciones del modelo están muy cerca de las posiciones reales de los objetos, lo que es crucial para aplicaciones donde la precisión espacial es importante.

- Cls Loss (Classification Loss): Se refiere al error en la clasificación de los objetos dentro de las cajas delimitadoras. Esta métrica evalúa la capacidad

del modelo para asignar correctamente las etiquetas de clase a los objetos detectados. Un valor bajo de Cls Loss indica que el modelo es eficaz en la clasificación precisa de los objetos, lo cual es esencial para aplicaciones donde la identificación correcta de la clase es crítica (Szegedy et al., 2015)[34].

- Dfl Loss (Distribution Focal Loss): Se utiliza para mejorar la calidad de la predicción de las cajas, ajustando las distribuciones de probabilidad de los puntos clave. Esta métrica se centra en mejorar la precisión de las predicciones de los bordes de las cajas delimitadoras mediante la utilización de una distribución focal que penaliza más los errores en los puntos clave de las cajas. Un valor bajo de Dfl Loss indica que el modelo está prediciendo con mayor precisión las ubicaciones exactas de los puntos clave, lo que es especialmente importante para la detección precisa en escenarios complejos y densamente poblados.

8.5.7. Velocidades de Procesamiento

Las velocidades de procesamiento, incluyendo el tiempo de pre-procesamiento, inferencia y post-procesamiento, son factores críticos que afectan la eficiencia total del sistema y su aplicabilidad en escenarios de tiempo real.

Tiempo de pre-procesamiento: Se refiere al tiempo requerido para preparar los datos de entrada antes de que sean procesados por el modelo. Esto puede incluir tareas como la redimensionamiento de imágenes, normalización de los datos y aplicación de técnicas de aumento de datos (data augmentation). Un tiempo de pre-procesamiento optimizado es esencial para asegurar que el sistema pueda manejar flujos de datos en tiempo real sin demoras significativas (Xu, C., Huang, Y., & Zhao, J. 2022)[35] .

Tiempo de inferencia: Es el tiempo que toma el modelo para procesar una imagen y generar predicciones. Este tiempo es crítico para aplicaciones en tiempo real, como la detección y conteo vehicular, donde se requiere una respuesta rápida y precisa para tomar decisiones inmediatas. Modelos más ligeros y optimizados, como lo es Yolov8n, están diseñados para reducir el tiempo de inferencia manteniendo una alta precisión, lo que los hace ideales para aplicaciones móviles y en dispositivos con recursos limitados.

8.5.8. Entropía Cruzada Categórica Dispersa

Es una métrica de pérdida ampliamente utilizada en tareas de clasificación con múltiples clases, particularmente cuando las etiquetas están representadas como enteros en lugar de vectores one-hot. Esta métrica mide la discrepancia entre la distribución de probabilidad predicha por el modelo y la distribución real, penalizando fuertemente las predicciones incorrectas.

Matemáticamente, la entropía cruzada categórica dispersa se define como:

$$L(y, \widehat{y}) = -\sum_{i=1}^{N} y_i * log(\widehat{y}_i)$$

Donde:

- y_i es la etiqueta verdadera para la clase ii,
- $\widehat{y}_i$ es la probabilidad predicha para la clase ii,
- N es el número total de clases.

En el caso de la entropía cruzada categórica dispersa, yy es un entero que representa la clase verdadera, en lugar de un vector one-hot. Esto simplifica el cálculo y ahorra memoria, especialmente útil en escenarios con muchas clases.

La interpretación de la entropía cruzada categórica dispersa es directa: un valor más bajo de la pérdida indica que las predicciones del modelo están más alineadas con las etiquetas verdaderas. En otras palabras, el modelo está haciendo predicciones más precisas (Goodfellow, Bengio & Courville, 2016)[23].

Para la detección y conteo vehicular, la entropía cruzada categórica dispersa se utiliza para optimizar la parte de clasificación del modelo, asegurando que cada objeto detectado se clasifique correctamente en una de las categorías predefinidas (por ejemplo, carro, moto, bus/camión).

Un valor de entropía cruzada categórica dispersa cercano a 0 es ideal, indicando que el modelo tiene una alta confianza en sus predicciones correctas. En contraste, valores altos sugieren que el modelo está teniendo dificultades para diferenciar entre clases, lo que puede deberse a datos de entrenamiento insuficientes, mal etiquetados o a una arquitectura de modelo inadecuada.

9. RESULTADOS Y DISCUSIÓN

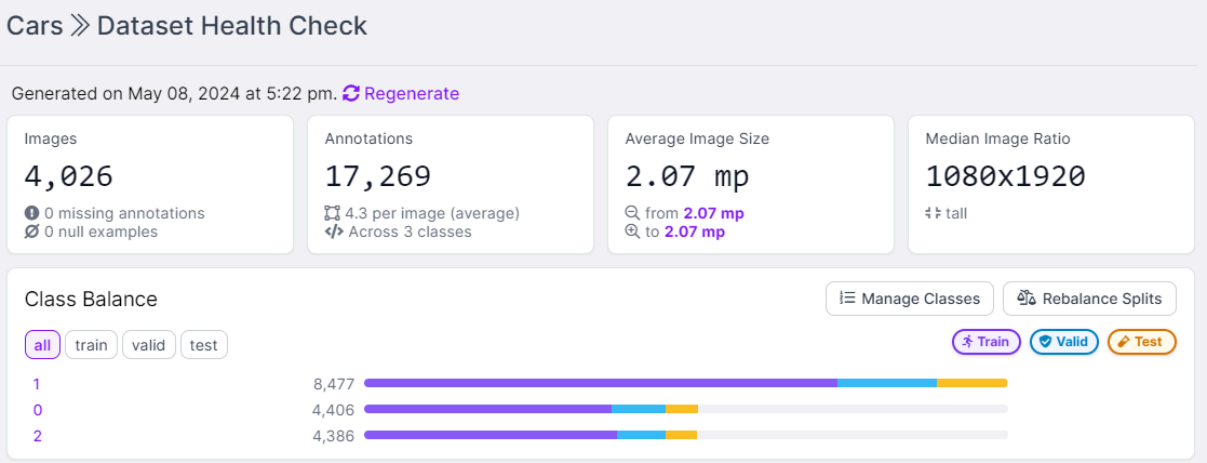
En esta sección se presentan los resultados obtenidos del modelo mejor entrenado, basado en un enfoque iterativo donde cada modelo fue entrenado sobre los pesos guardados del modelo anterior, en donde este método permitió superar las limitaciones computacionales existentes. Se discutirán algunas métricas relevantes ofrecidas por la arquitectura YOLO en cuanto a su evaluación cuantitativa. Además, se realiza una comparación entre la cantidad de vehículos contados de forma visual y la cantidad detectada por el modelo para obtener métricas más precisas sobre la precisión del modelo.

Para el entrenamiento del modelo final, se realizaron cinco iteraciones de entrenamiento sobre cada modelo obtenido previamente. Para cada iteración, se utilizaron aproximadamente 4000 imágenes distintas (incluyendo condiciones de iluminación variadas, lluvia, nubes y diferentes ángulos), con alrededor de cuatro etiquetas por imagen, sumando un total de 20,000 imágenes y 80,000 etiquetas. El modelo utilizado fue YOLOv8n, la versión más ligera y rápida, aunque con menor precisión. Por lo cual para superar estas limitaciones, se optó por utilizar una gran cantidad de datos y una serie iterativa de entrenamientos.

Este enfoque permitió no solo optimizar el rendimiento del modelo dentro de las restricciones computacionales, sino también mejorar la precisión y robustez del sistema de detección y conteo vehicular. Los resultados y las discusiones presentadas en esta sección reflejan la efectividad del enfoque adoptado y ofrecen una visión detallada del rendimiento del modelo en condiciones reales.

Figura 16

Interfaz de RoboFlow.



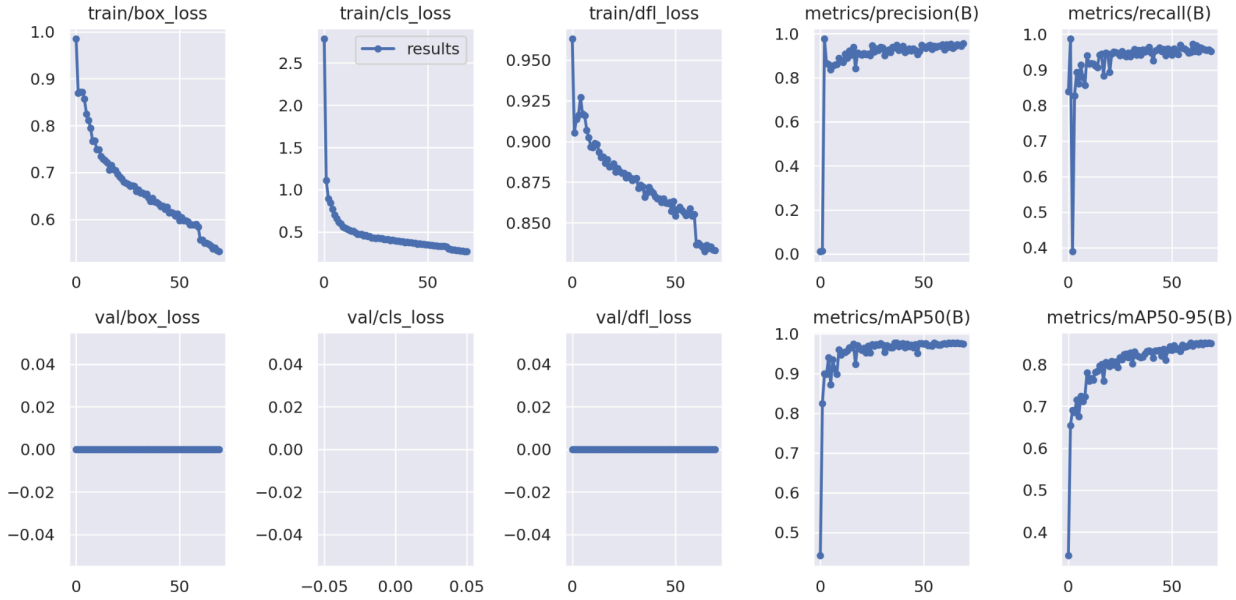
Nota. Por Autor, 2024. Ejemplo del número de anotaciones e imágenes de la iteración 1.

9.1. MÉTRICAS DE EVALUACIÓN

9.1.1. Curvas de pérdida (loss)

Figura 17

Curvas de pérdida (loss)



Nota. Por Autor, 2024. Gráficas de resultados del entrenamiento.

Box Loss:

Como se mencionó anteriormente el Box Loss miden el error en la predicción de las coordenadas de las cajas delimitadoras, afectando directamente la precisión de la localización. Durante las 69 épocas de entrenamiento, se observa una disminución constante de 0.98498 en la época 0 a 0.53159 en la época 69, lo que indica una mejora continua en la capacidad del modelo para localizar objetos con precisión. Un valor bajo de Box Loss sugiere que el modelo YOLOv8 es eficaz en la localización de vehículos, lo cual es esencial para el proyecto de aforo vehicular en tiempo real. Esta métrica es crucial ya que una localización precisa permite un conteo vehicular más fiable y una mejor gestión del tráfico.

Cls Loss:

La Cls Loss refleja el error en la clasificación de los objetos dentro de las cajas delimitadoras. A lo largo del entrenamiento, se observa una reducción significativa de 2.78570 en la época 0 a 0.26859 en la época 69, indicando mejoras en la precisión de clasificación. Un valor bajo de Cls Loss sugiere que el modelo está clasificando correctamente los tipos de vehículos, lo cual es fundamental para el conteo preciso de diferentes categorías de vehículos (carros, motos, buses/camiones) en el proyecto.

Dfl Loss:

La Distribution Focal Loss (Dfl Loss) se utiliza para mejorar la calidad de la predicción de las cajas, ajustando las distribuciones de probabilidad de los puntos clave. La Dfl Loss disminuye gradualmente de 0.96313 en la época 0 a 0.83295 en la época 69, lo que indica mejoras en la precisión de los bordes de las cajas delimitadoras. Esta métrica es relevante para el proyecto de aforo vehicular, ya que una mejor precisión en los bordes de las cajas permite una detección más precisa de los límites de cada vehículo, contribuyendo a una mejor segmentación y conteo en tiempo real.

Precisión:

La precisión mide la proporción de predicciones correctas entre todas las predicciones realizadas por el modelo. A lo largo de las épocas de entrenamiento, se observa un aumento constante, desde 0.01101 en la época 0 hasta 0.95629 en la época 69. Un alto valor de precisión sugiere que el modelo YOLOv8 realiza predicciones correctas de manera consistente, lo cual es crucial para el conteo vehicular en tiempo real. Una alta precisión asegura que la mayoría de las detecciones de vehículos sean correctas, reduciendo el número de falsos positivos y mejorando la fiabilidad del sistema de gestión del tráfico.

Recall:

El recall mide la proporción de verdaderos positivos entre todas las instancias reales. Se mantiene generalmente alto, con un leve descenso al final, comenzando en 0.83910 en la época 0 y terminando en 0.95182 en la época 69. Un alto valor de recall indica que el modelo está detectando la mayoría de los vehículos presentes en las imágenes. En el contexto de este proyecto, un alto recall es esencial para asegurar que se detecten todos los vehículos presentes, evitando subestimaciones en el conteo vehicular y permitiendo una gestión más precisa del tráfico.

mAP50 y mAP50-95:

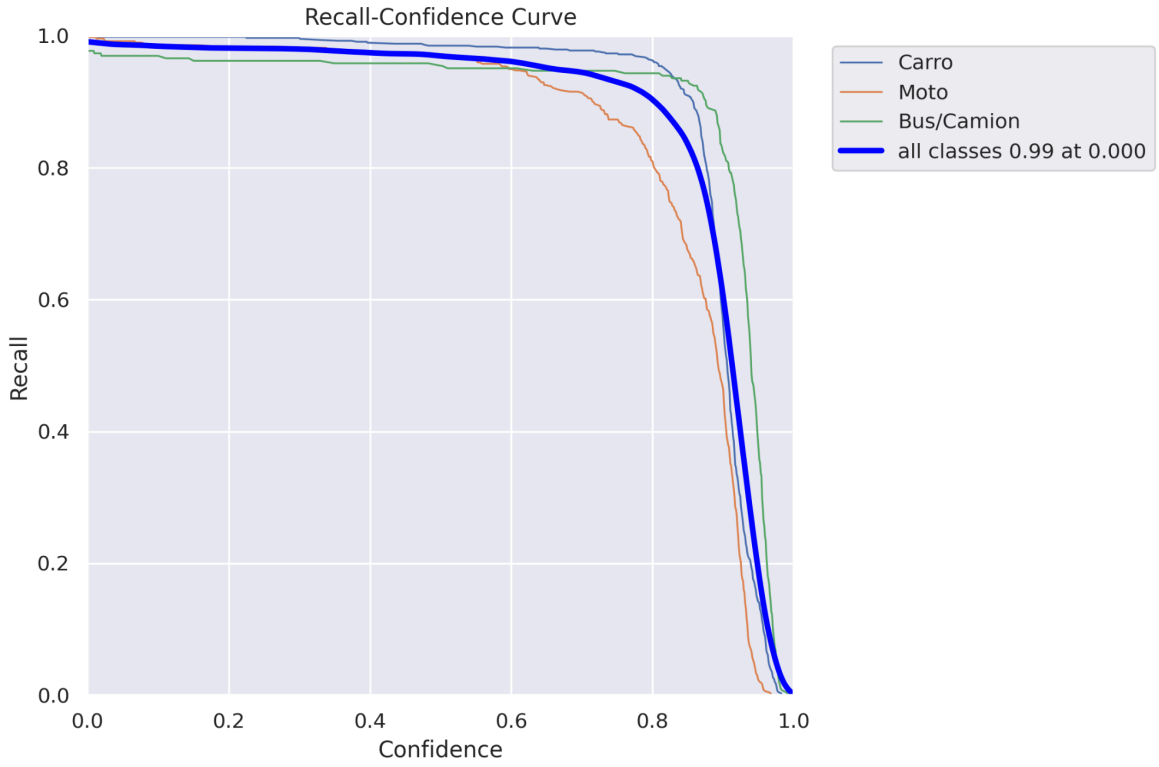
La Mean Average Precision a un umbral de IoU de 0.5 (mAP50) y entre 0.5 y 0.95 (mAP50-95) refleja mejoras en la precisión media del modelo. mAP50 aumenta de 0.44162 en la época 0 a 0.97423 en la época 69, y mAP50-95 de 0.34363 a 0.84978. Valores altos de mAP indican una buena precisión tanto en la localización como en la clasificación de objetos. Para este proyecto, esto significa que el modelo YOLOv8 no solo detecta y localiza correctamente los vehículos, sino que también los clasifica adecuadamente, proporcionando datos precisos y confiables para el conteo vehicular en tiempo real.

9.2. ANÁLISIS VISUAL DE LAS GRÁFICAS

9.2.1. Recall-Confidence Curve

Figura 18

Recall-Confidence Curve



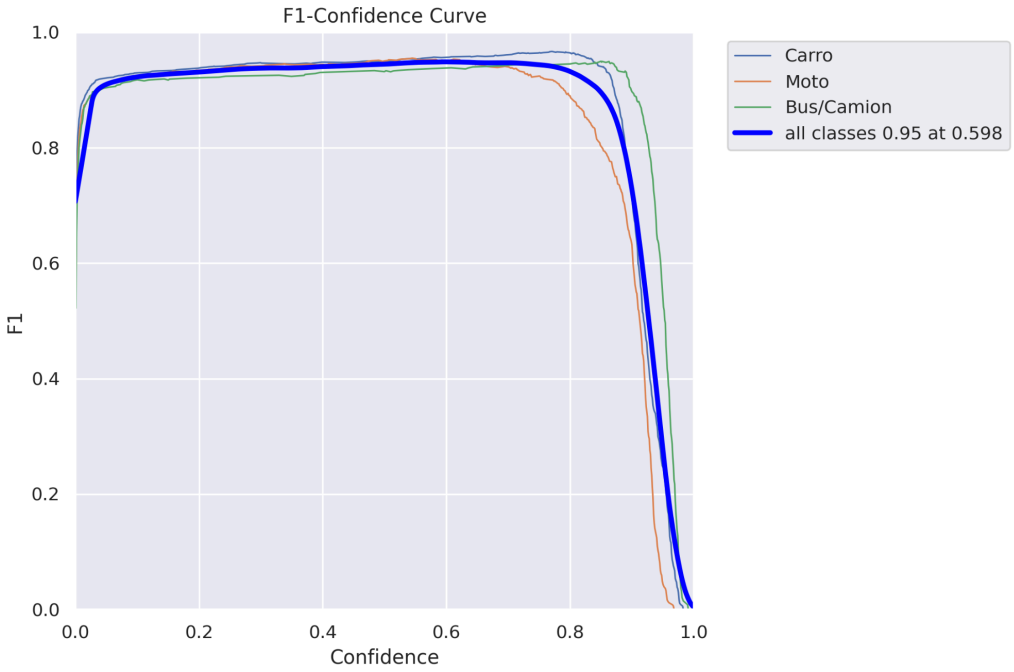
Nota. Por Autor, 2024.

La curva muestra que la mayoría de las clases mantienen un recall cercano a 1 hasta un umbral de confianza de aproximadamente 0.8, después del cual disminuye rápidamente. Esto indica que el modelo puede detectar la mayoría de los vehículos con alta confianza hasta cierto umbral, lo cual es ideal para ajustar el sistema de conteo vehicular en tiempo real para maximizar la detección sin comprometer la precisión.

9.2.2. F1-Confidence Curve

Figura 19

F1-Confidence Curve



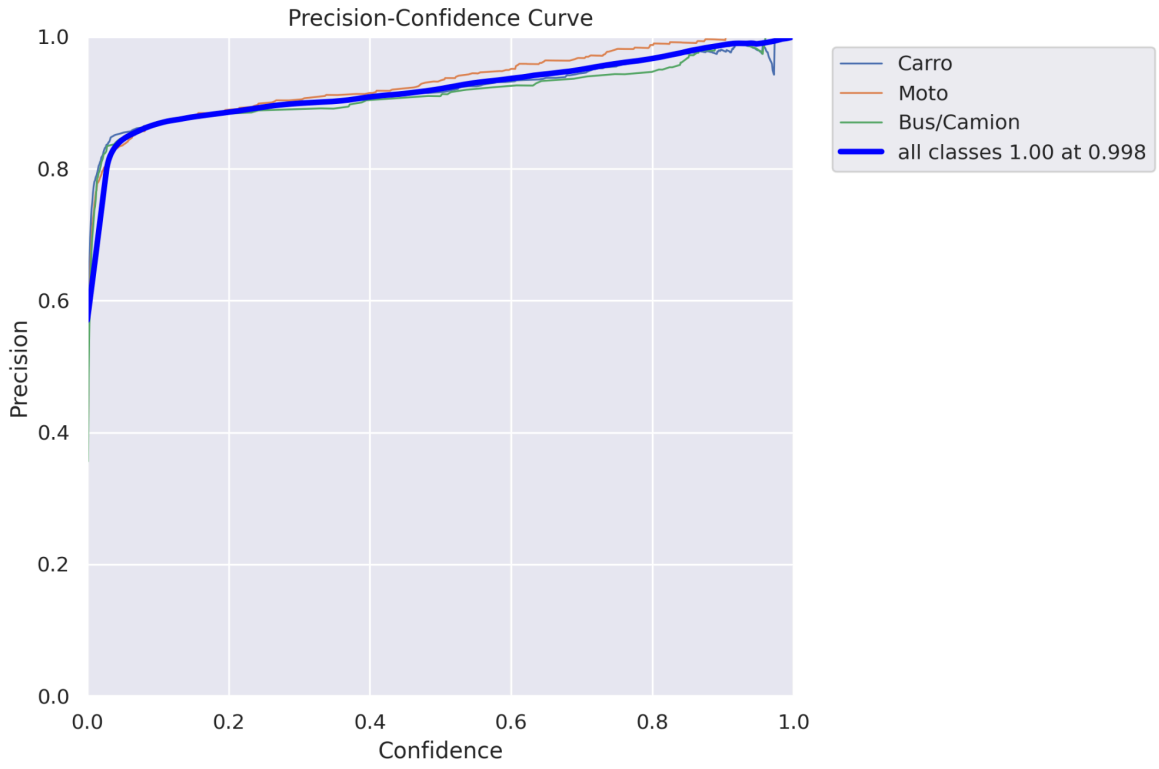
Nota. Por Autor, 2024.

Los valores de F1 Score son altos para todas las clases hasta un umbral de confianza de aproximadamente 0.6-0.8, sugiriendo un buen equilibrio entre precisión y recall. Este equilibrio es crucial para el proyecto, ya que permite un conteo vehicular preciso y fiable, minimizando tanto los falsos positivos como los falsos negativos.

9.2.3. Precision-Confidence Curve

Figura 20

Precision-Confidence *Curve*



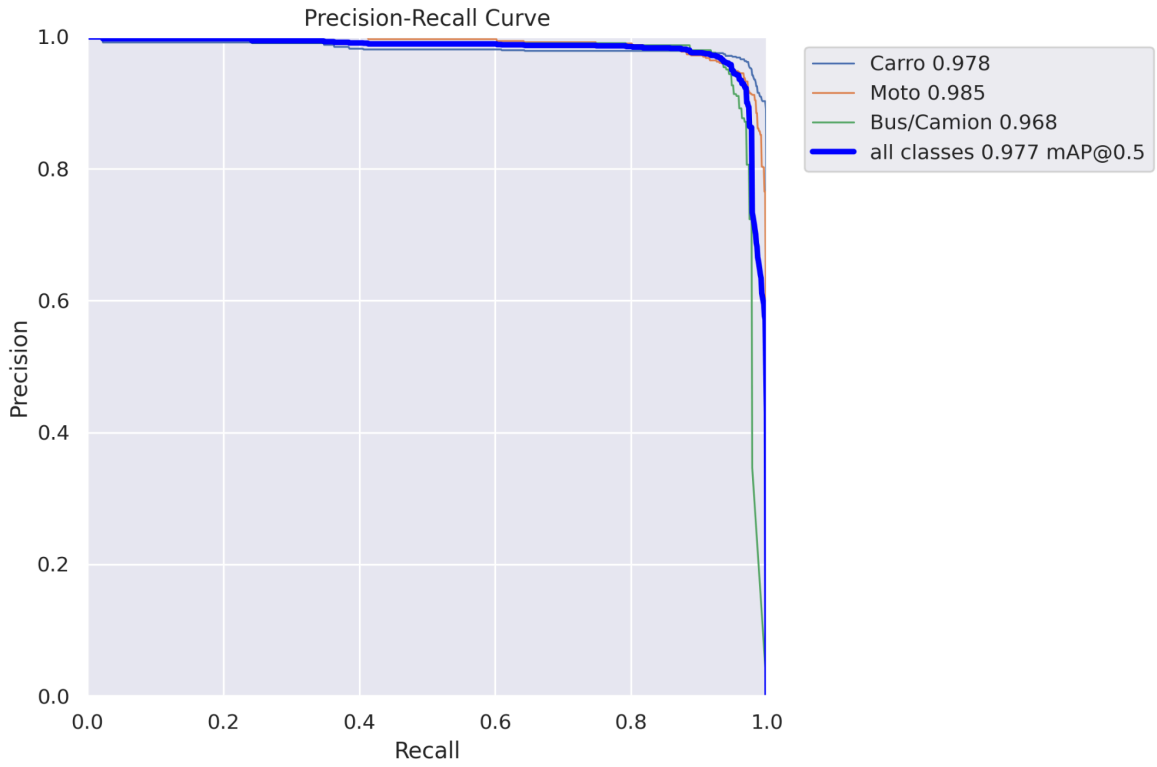
Nota. Por Autor, 2024.

La precisión aumenta con el umbral de confianza, lo que es esperado y deseable. Esto muestra que las predicciones con alta confianza son más precisas, lo cual es importante para asegurar que el sistema de conteo vehicular no haga falsas detecciones que podrían afectar la exactitud de los datos de tráfico.

9.2.4. Precision-Recall Curve

Figura 21

Precision-Recall Curve



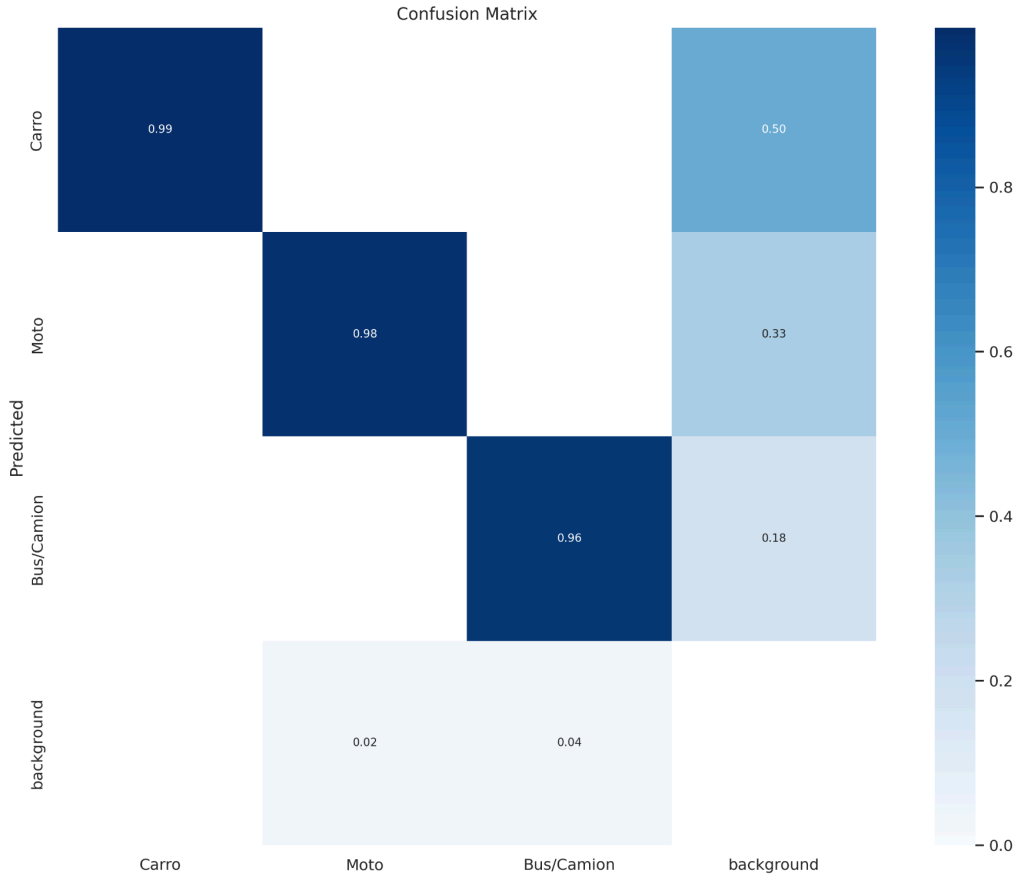
Nota. Por Autor, 2024.

Valores altos de precisión y recall para todas las clases son indicativos de un buen rendimiento general del modelo. Esto asegura que el sistema puede mantener un buen desempeño en ambas métricas clave, proporcionando datos de tráfico precisos y fiables.

9.2.5. Matriz de Confusión

Figura 22

Matriz de Confusión



Nota. Por Autor, 2024.

La matriz de confusión proporciona una representación visual de las predicciones correctas e incorrectas del modelo, facilitando la identificación de patrones de error entre diferentes clases.

Carro:

Verdaderos Positivos (TP): La celda de la diagonal principal (Carro-Carro) muestra un valor de 0.99, indicando que el modelo tiene una alta precisión en la detección de carros.

Falsos Positivos (FP) y Falsos Negativos (FN): Hay valores fuera de la diagonal para carros clasificados incorrectamente como otras clases, como 0.50 en la columna de fondo, sugiriendo algunos falsos positivos donde se detectaron carros en lugares incorrectos.

Moto:

Verdaderos Positivos (TP): La celda de la diagonal principal (Moto-Moto) muestra un valor de 0.98, lo que indica una alta precisión en la detección de motos.

Falsos Positivos (FP) y Falsos Negativos (FN): Los errores de clasificación para motos son menores, con valores como 0.33 en la columna de fondo, sugiriendo algunos falsos positivos.

Bus/Camión:

Verdaderos Positivos (TP): La celda de la diagonal principal Bus/Camión-Bus/Camión) muestra un valor de 0.96, indicando una buena precisión en la detección de buses/camiones.

Falsos Positivos (FP) y Falsos Negativos (FN): Hay errores de clasificación notables con valores como 0.18, indicando confusiones entre buses/camiones y otras clases.

Fondo (Background):

Verdaderos Negativos (TN): Los valores de fondo clasificados correctamente (celdas fuera de la diagonal principal) son bajos, indicando que el modelo está clasificando la mayoría de las veces correctamente las áreas que no contienen vehículos.

9.3. VALIDACIÓN

Para el análisis de tráfico vehicular y la gestión de movilidad, se han desarrollado varios códigos que aprovechan técnicas avanzadas de visión por computadora y aprendizaje automático. Estos códigos se enfocan en tareas como la detección, clasificación, conteo y cálculo de velocidades de vehículos, tanto a partir de vídeos procesados como en tiempo real. A continuación, se describen dos enfoques principales:

9.3.1. Video Procesado

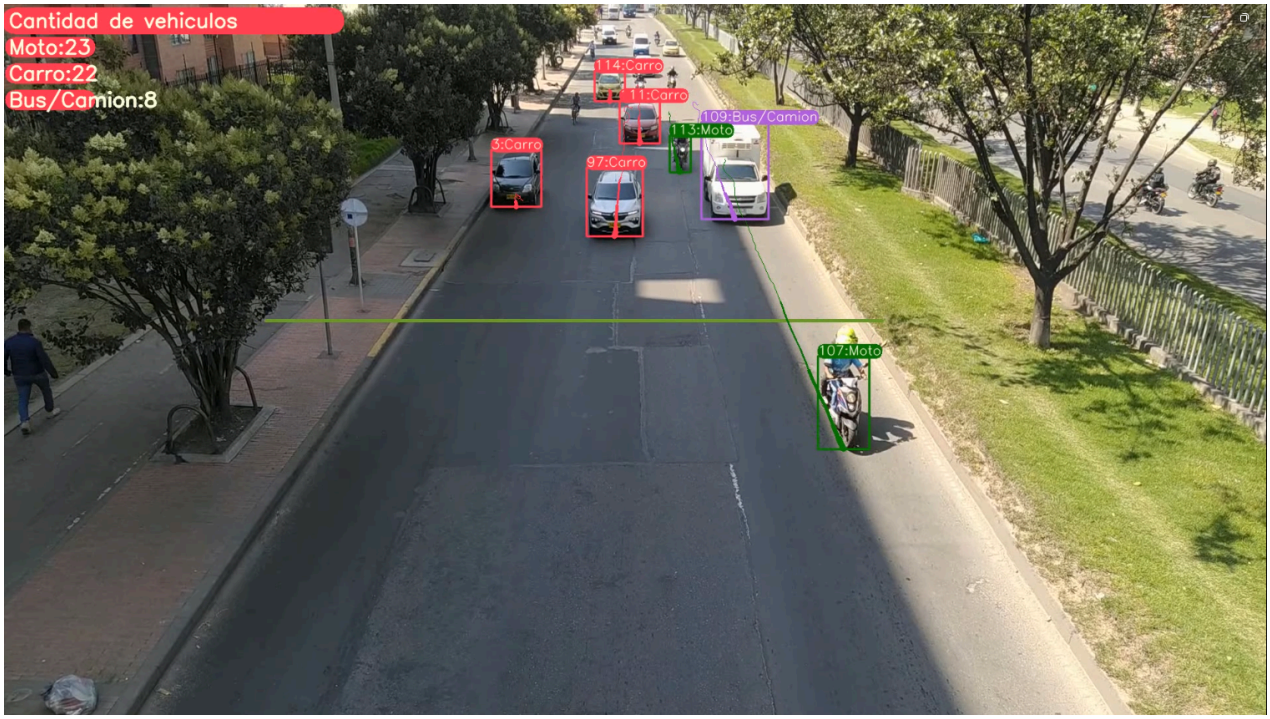
Se realizó la evaluación del modelo YOLOv8 con el objetivo de determinar su efectividad en la detección, conteo y cálculo de velocidades de vehículos al procesar un video de tráfico vehicular. Para llevar a cabo esta prueba, se utilizó un video grabado en la avenida Ciudad de Villavicencio con la transversal 70, en el barrio Madelena de la localidad 19 (Ciudad Bolívar), Bogotá, Colombia. El material audiovisual empleado presentaba un escenario con un total de 31 carros, 31 motos y 12 buses/camiones circulando. Tras el procesamiento del video por parte del modelo YOLOv8, los resultados obtenidos en cuanto a detección, conteo y velocidades calculadas fueron comparados con los valores previamente conocidos de la composición vehicular.

Código 1: Detección, Clasificación y Conteo de Vehículos con Generación de Gráficas.

Este código implementa un sistema integral para la detección, clasificación, conteo y análisis de vehículos a partir de un flujo de video en tiempo real. Utiliza un modelo YOLO para detectar vehículos y clasificarlos en categorías como carros, motos y buses/camiones. Luego, emplea el algoritmo Deep SORT para realizar el rastreo de cada vehículo detectado a lo largo de múltiples fotogramas. Se calcula la dirección de desplazamiento de los vehículos para determinar si cruzan una línea de detección predefinida, registrando la categoría del vehículo y la hora en que se realiza cada conteo al cruzar dicha línea. Al finalizar el procesamiento del video, se genera una gráfica que muestra el número de vehículos detectados por categoría, anotando las horas de inicio y fin del análisis para proporcionar un contexto temporal claro. En donde se obtuvieron los resultados de 30 carros, 30 motos y 14 camiones. Se adjunta una gráfica generada por el código mostrando los resultados obtenidos por el modelo.

Figura 23

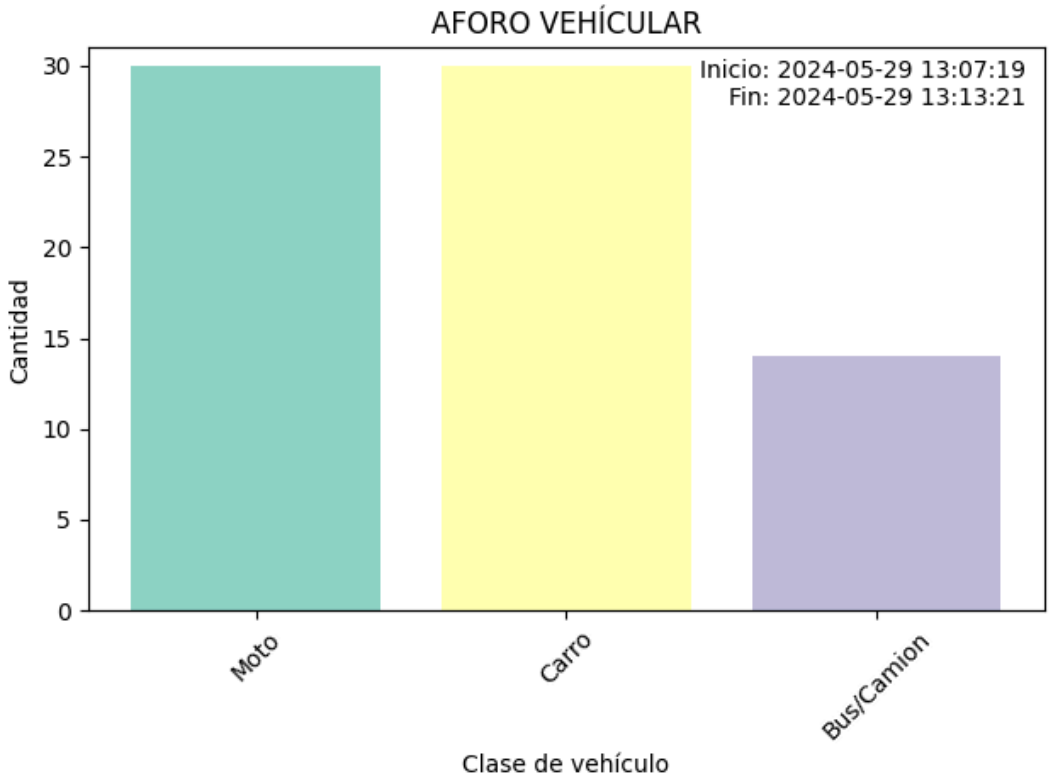
Resultados código 1



Nota. Por Autor, 2024.

Figura 24

Diagrama de barras código 1



Nota. Por Autor, 2024. En el diagrama de barras se recopila toda la información procesada y muestra la cantidad y clases de vehículos que pasan en ese lapso de tiempo definido.

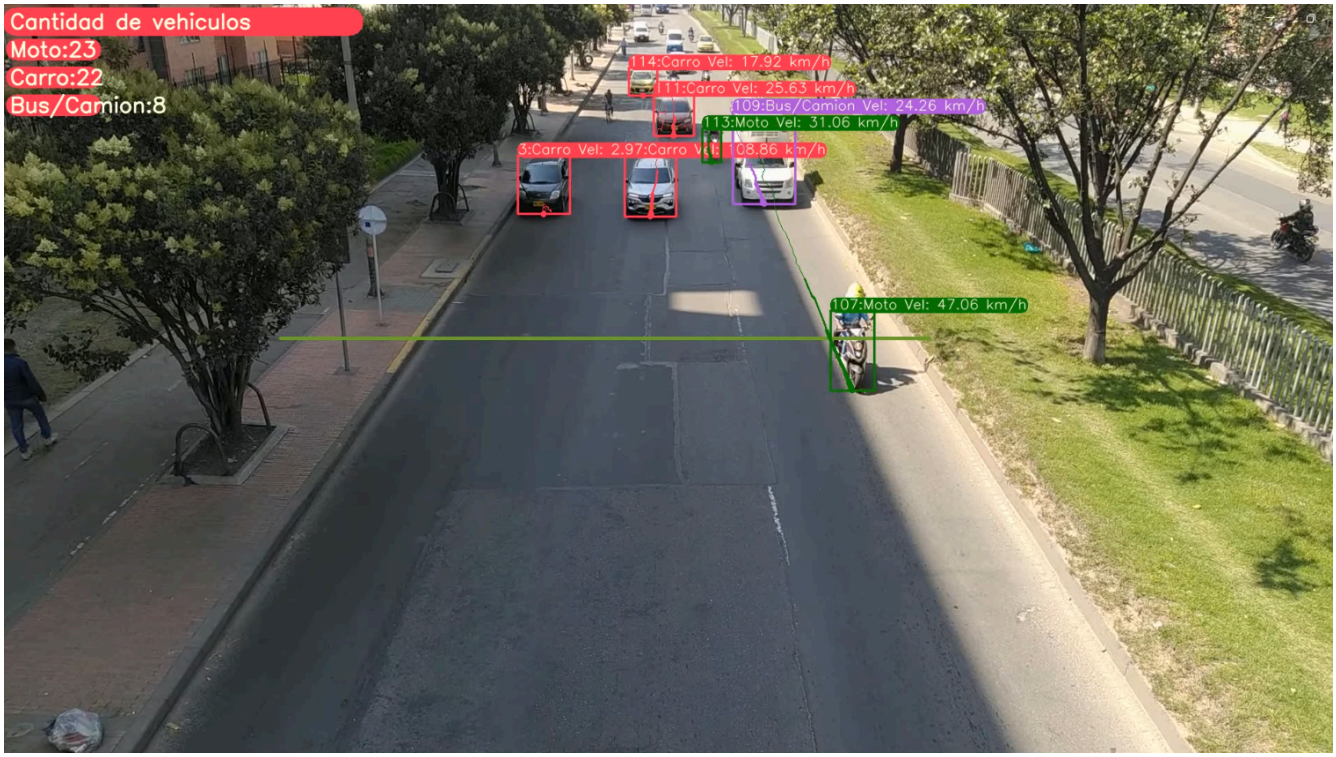
Código 2: Cálculo de velocidad para cada clase con Generación de Gráficas.

Este código implementa un sistema avanzado para detectar y rastrear objetos en un video, calculando su velocidad a través del análisis del desplazamiento de las cajas delimitadoras en píxeles por fotograma. Utiliza algoritmos de detección de objetos para identificarlos y generar cajas delimitadoras alrededor de ellos, monitoreando su posición en el tiempo. Calcula el desplazamiento del centro del borde inferior de estas cajas entre fotogramas consecutivos, convirtiendo los

píxeles a distancia real mediante un factor de conversión basado en las dimensiones del escenario. Luego, aplica la fórmula de velocidad utilizando la distancia en kilómetros y el tiempo entre fotogramas para obtener la velocidad en km/h. Las velocidades se muestran en tiempo real en el video y se registran cuando los objetos cruzan una línea de detección predefinida. En la implementación de este código, se obtienen los mismos resultados que en el código usado anteriormente: 30 carros, 30 motos y 14 camiones. Al finalizar, genera una gráfica que muestra la cantidad de vehículos detectados y sus velocidades promedio por clase, con anotaciones detalladas, facilitando el análisis visual y la interpretación de los datos de tráfico y monitoreo. Se adjunta una gráfica generada por el código mostrando los resultados obtenidos por el modelo.

Figura 25

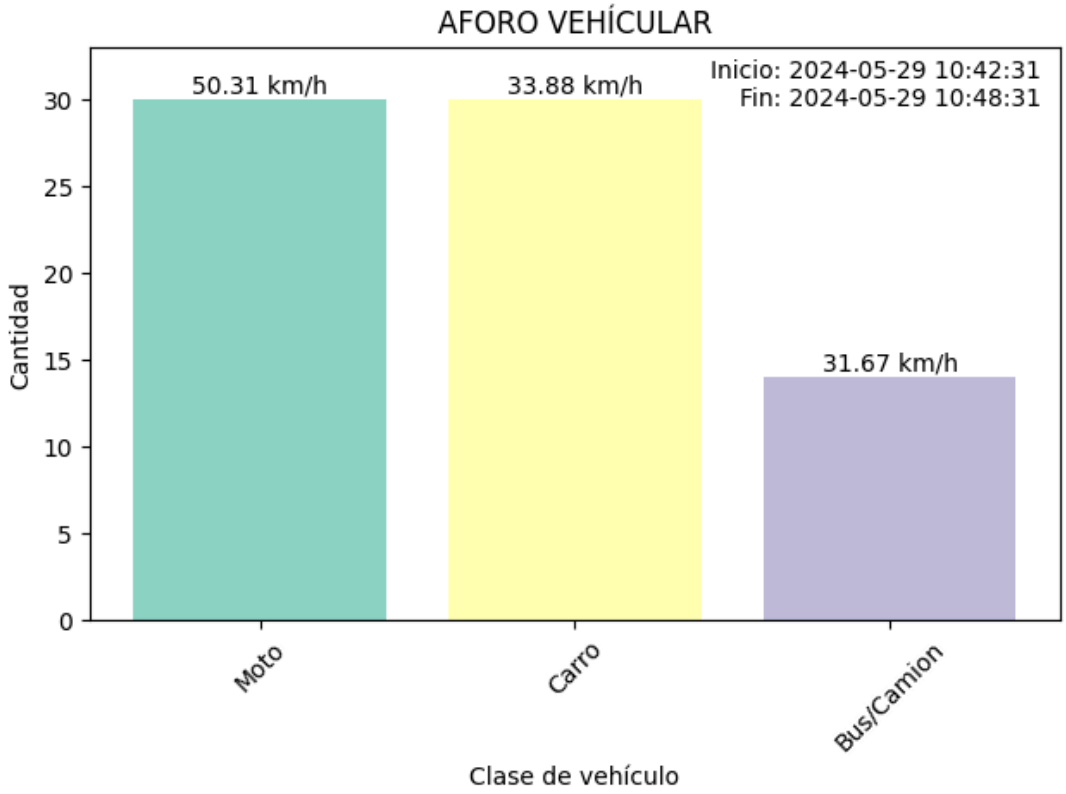
Resultados código 2



Nota. Por Autor, 2024.

Figura 26

Diagrama de barras código 2



Nota. Por Autor, 2024. Arroja exactamente los mismos valores que en la figura 24, pero a diferencia, esta muestra las velocidades promedio para cada clase.

9.3.2. Tiempo Real

Se realizó una prueba del modelo YOLOv8 para evaluar su capacidad de detectar y contar vehículos en tiempo real en la avenida Ciudad de Villavicencio con la transversal 70, en el barrio Madelena de la localidad 19 (Ciudad Bolívar), Bogotá, Colombia. La prueba se efectuó proyectando un video en un televisor y capturando la transmisión mediante la cámara de un celular, que luego procesó la información en tiempo real. El video contenía 13 carros, 23 motos y 7

buses/camiones. Los resultados del modelo fueron comparados con estos valores para evaluar su precisión.

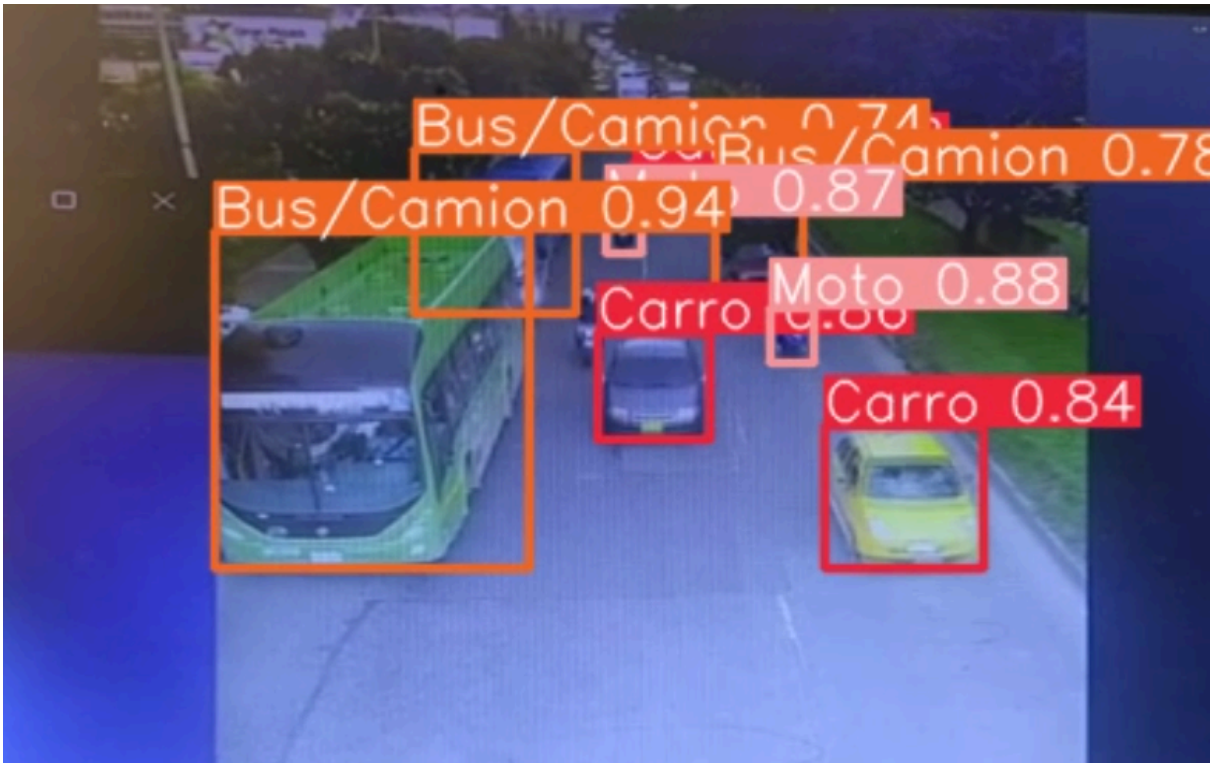
Debido a limitaciones computacionales, el procesamiento simultáneo de múltiples funciones generó una alta demanda en la GPU y la CPU del PC, lo que resultó en algunas imprecisiones. Para abordar esto, se probaron dos códigos:

Código 3: Detección Precisa de Vehículos con YOLOv8

Este código implementa la detección precisa de vehículos en tiempo real utilizando el modelo YOLO (You Only Look Once), enfocándose en identificar y clasificar diferentes tipos de vehículos como carros, motos y buses/camiones. Se configuró un entorno de captura con OpenCV para obtener un video de 640x480 píxeles de resolución, luego se cargó el modelo YOLO preentrenado y ajustado específicamente para detección de vehículos. En el procesamiento del video en tiempo real, se capturaron fotogramas continuos que se procesaron a través del modelo, identificando y clasificando los vehículos presentes, proporcionando las coordenadas de las cajas delimitadoras y las etiquetas de las clases detectadas. Los fotogramas anotados se visualizaron en una ventana de video utilizando cv2.imshow. En donde el modelo logra identificar con precisión el 100% de los vehículos, detectando 13 carros, 23 motos y 7 buses/camiones, demostrando su alta eficacia y rapidez en tareas de detección de vehículos en tiempo real.

Figura 27

Resultados código 3



Nota. Por Autor, 2024.

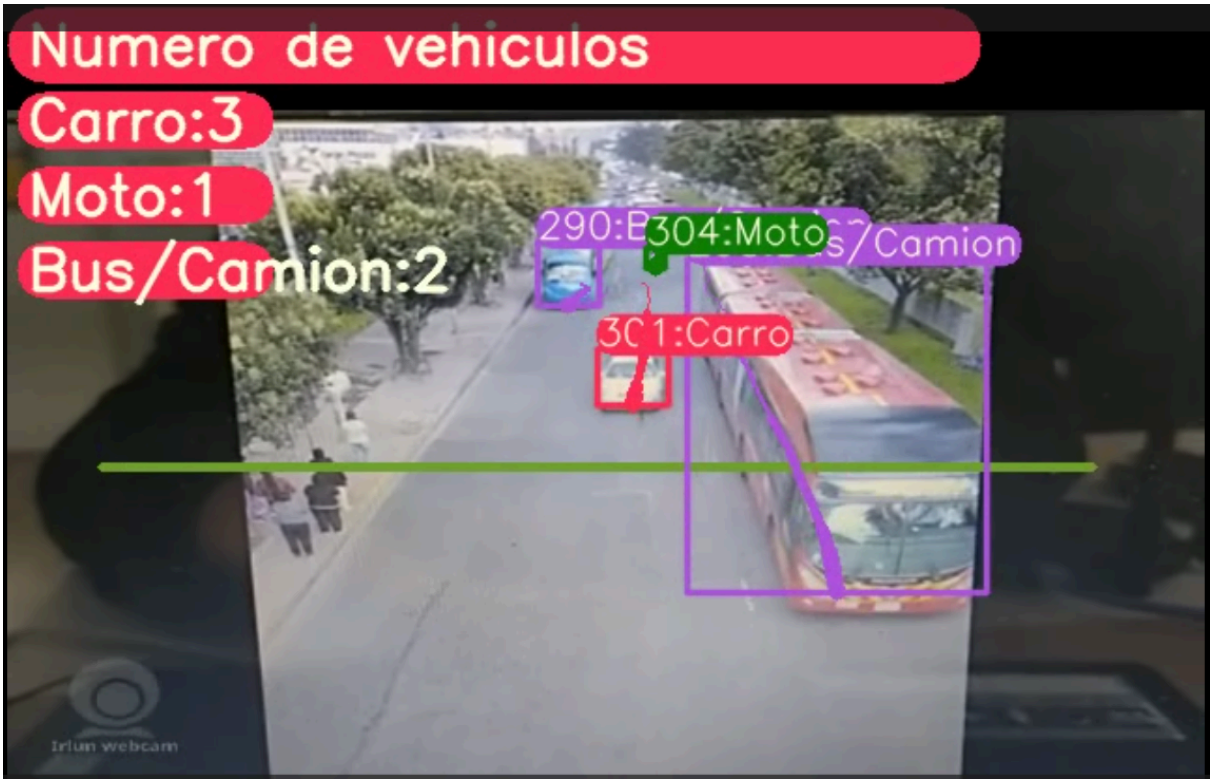
Código 4: Detección, Clasificación y Conteo de Vehículos en Tiempo Real con Generación de Gráficas en Tiempo Real:

Este código tiene las mismas características del Código 1, pero en lugar de procesar un video grabado previamente, se enfoca en el análisis de una fuente de video en vivo, como una cámara de celular. Aprovechando las capacidades del modelo YOLO y el algoritmo Deep SORT, el sistema realiza la detección, clasificación y rastreo de vehículos en tiempo real, identificando categorías como carros, motos y buses/camiones. A medida que los vehículos cruzan una línea de detección predefinida, se registra su categoría y la hora del conteo.

Simultáneamente, se genera y actualiza una gráfica que muestra el número de vehículos detectados por categoría, proporcionando una visualización en tiempo real de los datos recopilados. Esta implementación en tiempo real permite un monitoreo continuo y un análisis instantáneo de la situación del tráfico. Sin embargo, aunque el modelo mostró alta precisión en la detección y clasificación, las limitaciones computacionales afectaron su rendimiento completo, detectando 11 carros, 13 motos y 7 buses/camiones, sin lograr contar y clasificar todos los vehículos con absoluta precisión. Se adjunta una gráfica generada por el código mostrando los resultados obtenidos por el modelo.

Figura 28

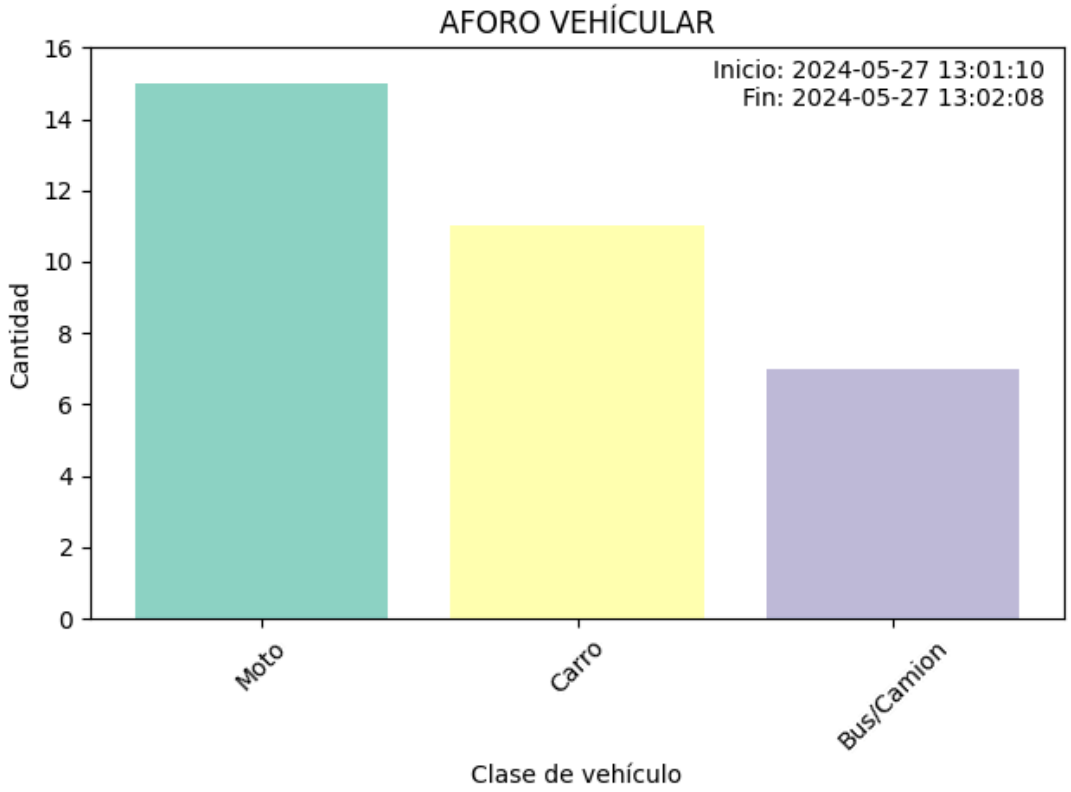
Resultados código 4



Nota. Por Autor, 2024.

Figura 29

Diagrama de barras código 4



Nota. Por Autor, 2024. En el diagrama de barras se recopila toda la información procesada y muestra la cantidad y clases de vehículos que pasan en ese lapso de tiempo definido.

A pesar de las limitaciones encontradas, estos resultados indican que, con mejoras en el hardware, el modelo YOLOv8 tiene el potencial de ser una herramienta eficaz para la detección y conteo de vehículos en tiempo real.

Nota: Todos los códigos implementados se encuentran en el apéndice A.

9.4. Predicción

Una vez obtenidos los resultados de los diferentes códigos y enfoques implementados, resulta indispensable llevar a cabo un riguroso análisis comparativo entre los datos arrojados y los valores reales conocidos. Esto para resaltar aspectos cruciales que se escapan de la representación gráfica obtenida, pero que son fundamentales para evaluar la verdadera precisión del modelo y código en cada enfoque.

En primer lugar, se comparan los recuentos vehiculares discriminados por categoría (carros, motos, camiones/buses) con las cantidades reales presentes en el video de prueba. Esto revelará cualquier discrepancia o sesgo en la capacidad de detección y clasificación de los modelos empleados. Además, se debe prestar especial atención a los casos de falsos positivos o negativos, donde vehículos hayan sido erróneamente identificados o pasados por alto. Por otro lado, en cuanto al cálculo de velocidades, será necesario confrontar los valores obtenidos por el algoritmo con mediciones de referencia.

Video Procesado

Código 1 y 2: Detección, Clasificación, Conteo y velocidad promedio de Vehículos con Generación de Gráficas.

Al analizar los resultados de detección obtenidos por el código, se evidencian ciertas discrepancias con respecto a los datos reales. Según se mencionó previamente, el código detectó 30 carros, 30 motos y 14 camiones. Sin embargo, los valores verdaderos correspondientes al video procesado eran 31 carros, 31 motos y 12 buses/camiones circulando.

Tras revisar detenidamente la grabación utilizada para el Código 1, se pudo constatar que el modelo no omitió ningún vehículo presente, descartando así la existencia de falsos negativos totales. No obstante, se identificaron varios casos de falsos negativos parciales, donde el modelo clasificó erróneamente ciertos

vehículos en categorías incorrectas. Específicamente, se observó que dos carros fueron erróneamente detectados como buses/camiones, un bus/camión fue identificado como un carro, y una moto fue catalogada como un bus/camión. Estas confusiones en la clasificación pueden atribuirse a diversos factores, como la perspectiva de la cámara, las condiciones de iluminación o la superposición parcial de vehículos en ciertas escenas. Es importante destacar que, a pesar de estas discrepancias en la clasificación, el modelo no registró ningún falso positivo, es decir, no identificó erróneamente objetos que no fueran vehículos como tales. Esta es una fortaleza significativa, ya que los falsos positivos pueden generar ruido y distorsionar los recuentos vehiculares.

Carros:

- Total de carros presentes: 31
- Carros detectados correctamente: 29
- Precisión de detección de carros:

$$\frac{Carros\ detectados\ correctamente}{Total\ de\ carros\ presentes} = \frac{29}{31} = 0.935 = 93.5\%$$

Motos:

- Total de motos presentes: 31
- Motos detectadas correctamente: 30
- Precisión de detección de motos:

$$\frac{Motos\ \ detectadas\ correctamente}{Total\ de\ motos\ presentes} = \frac{30}{31} = 0.968 = 96.8\%$$

Buses/Camiones:

- Total de buses/camiones presentes: 12
- Buses/camiones detectados correctamente: 11

- Precisión de detección de buses/camiones:

$$\frac{Buses/Camiones\ detectados\ correctamente}{Total\ de\ Buses/Camiones\ presentes} = \frac{11}{12} = 0.917 = 91.7\%$$

Precisión general

$$\frac{Vehículos\ detectados\ correctamente}{Total\ vehículos\ presentes} = \frac{70}{74} = 0.946 = 94.6\%$$

Teniendo en cuenta los errores de clasificación observados, se puede calcular que el modelo alcanzó una precisión general de detección del 93.5% para carros, 96.8% para motos y 91.7% para buses/camiones. Estos valores, si bien no son perfectos, representan un rendimiento aceptable considerando la complejidad de la tarea y las condiciones del entorno.

Por otro lado, al comparar las velocidades obtenidas por el código con las mediciones manuales calculadas a partir de la ecuación $v = \frac{d}{t}$, se pudo constatar que el sistema alcanzó una alta precisión en la estimación de las velocidades promedio de los vehículos detectados.

Carros:

- Velocidad real: 35 Km/h
- Velocidad promedio detectada: 33.8 Km/h
- Precisión de velocidad:

$$\frac{Velocidad\ promedio\ detectada}{Velocidad\ real} = \frac{33.8\ Km/h}{35\ Km/h} = 0.966 = 96.6\%$$

Motos:

- Velocidad real: 55 Km/h
- Velocidad promedio detectada: 50.1 Km/h
- Precisión de velocidad:

$$\frac{Velocidad\ promedio\ detectada}{Velocidad\ real} = \frac{50.1\ Km/h}{55\ Km/h} = 0.91 = 91\%$$

Buses/Camiones:

- Velocidad real: 32
- Velocidad promedio detectada: 31.67 Km/h
- Precisión de velocidad:

$$\frac{Velocidad\ promedio\ detectada}{Velocidad\ real} = \frac{31.67\ Km/h}{32\ Km/h} = 0.989 = 98.9\%$$

El modelo demostró un desempeño excepcional en el cálculo de las velocidades promedio de los vehículos detectados. Para los carros, se obtuvo una precisión del 96.6%, con una velocidad promedio detectada de 33.8 km/h frente a la velocidad real de 35 km/h. En cuanto a las motos, la precisión fue del 91%, con una velocidad promedio detectada de 50.1km/h en comparación con la velocidad real de 55 km/h. Finalmente, para los buses/camiones, el modelo alcanzó una impresionante precisión del 98.9%, estimando una velocidad promedio de 31.67 km/h muy cercana a la velocidad real de 32 km/h.

Estos resultados destacan la capacidad del sistema para realizar estimaciones de velocidad altamente precisas para diferentes tipos de vehículos, lo cual es fundamental en aplicaciones de monitoreo de tráfico y seguridad vial. Sin embargo, es importante tener en cuenta que aún existen márgenes de mejora,

especialmente en la detección y clasificación precisa de los vehículos, como se evidenció en el análisis de precisión anterior.

Para mejorar aún más el desempeño del modelo, se podrían explorar estrategias como el ajuste fino (fine-tuning) de los pesos de la red neuronal con un conjunto de datos más representativo del escenario específico, o utilizar un modelo con mayores pesos de YOLOv8, lo que podría mejorar la capacidad de discriminación entre las diferentes categorías vehiculares.

Tiempo Real

Código 3: Detección Precisa de Vehículos con YOLO

En las condiciones específicas bajo las cuales se realizó la evaluación, el modelo logró detectar con éxito la totalidad de los vehículos presentes en la escena, sin omitir ninguno, detectando 13 carros, 23 motos y 7 buses/camiones. Este resultado representa un logro significativo, ya que demuestra la capacidad del sistema para identificar de manera confiable la presencia de vehículos en el entorno analizado.

Al alcanzar una tasa de detección del 100% en este escenario particular, el modelo ha demostrado un alto nivel de confiabilidad y robustez en su capacidad de reconocimiento de objetos vehiculares. Esto es especialmente relevante en aplicaciones críticas, como el monitoreo de tráfico en tiempo real o el análisis forense de incidentes viales, donde la omisión de vehículos podría tener consecuencias graves.

Sin embargo, es crucial tener en cuenta que este rendimiento excepcional se obtuvo bajo condiciones específicas de iluminación, ángulo de cámara, densidad de tráfico y características del entorno. Es posible que al enfrentar escenarios

significativamente diferentes, el modelo pueda experimentar cierto grado de degradación en su tasa de detección.

Código 4: Detección, Clasificación y Conteo de Vehículos en Tiempo Real con Generación de Gráficas en tiempo real:

Es importante destacar que la cantidad de funciones a realizar en este código era significativa, lo que exigía una gran capacidad de procesamiento, esto representó una limitación en la calidad de los resultados obtenidos.

Este código en tiempo real fue capaz de identificar con éxito todos los vehículos presentes, pero cometió algunos errores de clasificación al etiquetar incorrectamente dos buses/camiones como motos, y dos carros como buses/camiones. Además, aunque detectó todos los presentes vehículos, no logró registrar apropiadamente su conteo en las categorías correspondientes.

Debido a las limitaciones de procesamiento que enfrentaba, el código no pudo evaluarse en su mayor capacidad y realizar con total precisión todas las tareas requeridas, como la clasificación y conteo preciso de los vehículos. Sin embargo, es importante resaltar que sí logró detectar con éxito la presencia de vehículos en la escena, lo cual es un avance significativo en sí mismo.

Sería interesante probar este código en una máquina con mayor capacidad de procesamiento, ya que podría mejorar su desempeño al tener recursos adicionales para realizar tareas complejas como la clasificación y el conteo preciso de vehículos. Asimismo, sería valioso evaluar su eficacia en escenarios más complicados, con condiciones más heterogéneas y un mayor número de objetivos a detectar y procesar, aprovechando su capacidad para detectar la presencia de vehículos de forma confiable.

10. CONCLUSIONES

10.1. Eficiencia en la Detección y Clasificación: El modelo YOLOv8 ha demostrado una alta eficiencia en la detección, conteo y clasificación precisa de distintos tipos de vehículos en imágenes de tráfico en tiempo real, alcanzando una notable precisión de hasta 95.6% y un recall de 95.2% al finalizar el entrenamiento. Esto indica que el modelo es capaz de identificar correctamente la mayoría de los vehículos presentes en las imágenes, cumpliendo con las expectativas iniciales.

10.2. Al evaluar el modelo con videos pregrabados, se detectaron 30 carros, 30 motos y 14 buses/camiones, en comparación con los valores reales de 31 carros, 31 motos y 12 buses/camiones. Estas pequeñas discrepancias evidencian la capacidad del modelo para manejar datos reales y diversos, lo cual es esencial para su desempeño en diversas condiciones de iluminación, ángulo de visión y tráfico en tiempo real, en donde la precisión general de detección fue del 94,6%.

10.3. Durante las pruebas en tiempo real, el modelo mostró un rendimiento robusto al detectar correctamente 13 carros, 23 motos y 7 buses/camiones, logrando una tasa de detección del 100% bajo las condiciones específicas del escenario evaluado. Sin embargo, las limitaciones computacionales afectaron la precisión de la clasificación y conteo de vehículos en escenarios más complejos. En pruebas adicionales con múltiples funciones simultáneas, se detectaron errores de

clasificación, como la identificación incorrecta de buses/camiones como motos y carros como buses/camiones. Estos resultados destacan la necesidad de mejorar la infraestructura computacional para optimizar el rendimiento del sistema en aplicaciones de tiempo real en condiciones reales de Bogotá.

10.4. Las métricas de pérdida (Box Loss, Cls Loss y Dfl Loss) mostraron una disminución significativa a lo largo del entrenamiento, reflejando mejoras en la capacidad del modelo para localizar y clasificar vehículos con precisión. Estas mejoras son indicativas de la eficiencia del modelo en la reducción de errores y el aumento de la precisión, alineándose con el objetivo de desarrollar un sistema de conteo y clasificación automatizada de vehículos.

10.5. La matriz de confusión mostró una alta tasa de verdaderos positivos para todas las clases de vehículos, con una precisión del 99% para carros, 98% para motos y 96% para buses/camiones. No obstante, se identificaron áreas de mejora en la clasificación de buses/camiones, lo que sugiere la necesidad de ajustes adicionales en los parámetros del modelo para mejorar su rendimiento.

10.6. El modelo demostró un desempeño excepcional en el cálculo de las velocidades promedio de los vehículos detectados, alcanzando una precisión del 96.6% para carros, 91% para motos y 98.9% para

buses/camiones. Estos resultados destacan la capacidad del sistema para estimar con alta precisión las velocidades de diferentes tipos de vehículos.

10.7. Para optimizar el rendimiento del sistema, se recomienda utilizar hardware más potente y optimizar el código para minimizar las imprecisiones y mejorar la eficiencia. Además, tambien se podría utilizar un modelo con mayores pesos de YOLOv8, lo que podría mejorar la capacidad de discriminación entre las diferentes categorías vehiculares.

10.8. Los resultados obtenidos indican que el sistema desarrollado tiene un alto potencial para ser implementado en la gestión del tráfico urbano, proporcionando datos precisos y en tiempo real que pueden mejorar la planificación del tráfico, reducir los tiempos de viaje y las emisiones contaminantes, y facilitar la toma de decisiones informadas en movilidad.

11. RECOMENDACIONES

Recomendaciones para Implementar un Proyecto de Aforo Vehicular Mediante Visión Artificial.

General:

Uso de un trípode: Durante la captura de imágenes o videos, se recomienda utilizar un trípode para evitar la distorsión de los fotogramas y asegurar una calidad constante.

Formato de imagen: Es recomendable capturar imágenes en formato 1:1 para evitar distorsiones significativas de los píxeles y asegurar una correcta proporción en el análisis.

Potencia de la máquina: Se sugiere ejecutar los códigos en máquinas potentes con capacidad suficiente de procesamiento y memoria para manejar grandes conjuntos de datos y modelos complejos.

Recomendaciones Técnicas:

Preprocesamiento de Imágenes: Se deben aplicar técnicas de preprocesamiento adecuadas, como normalización y corrección de color, para mejorar la calidad de las imágenes antes del etiquetado. Esto mejora la consistencia y la precisión de las detecciones.

Aumentación de Datos: Es fundamental implementar técnicas de aumentación de datos (como rotaciones, cambios de escala y ajustes en iluminación) para generar variaciones en las imágenes etiquetadas, lo que ayudará a mejorar la robustez y generalización del modelo.

Balanceo de Clases: Se debe verificar que las clases estén balanceadas en el conjunto de datos para evitar sesgos en el modelo. Si es necesario, se pueden aplicar técnicas de sobremuestreo o submuestreo.
Revisión de Etiquetas: Se recomienda realizar una revisión exhaustiva de las etiquetas para asegurar que sean precisas y consistentes. Para ello, se pueden utilizar herramientas de validación automática como Roboflow.

Recomendaciones de Hardware y Software:
Hardware Adecuado: Es esencial utilizar GPUs de alto rendimiento para entrenar los modelos de visión artificial, ya que esto reducirá significativamente el tiempo de entrenamiento.
Software y Bibliotecas: Se debe asegurar el uso de las versiones más actualizadas de las bibliotecas y frameworks, como TensorFlow, PyTorch y OpenCV, para aprovechar las últimas mejoras y correcciones de errores.
Entornos de Desarrollo: Se considera útil el uso de entornos de desarrollo en la nube, como Google Colab o AWS SageMaker, que ofrecen recursos computacionales escalables y facilitan la colaboración.

Recomendaciones de Captura y Calidad de Datos:

Condiciones de Iluminación: Es recomendable capturar imágenes en condiciones de iluminación controladas para reducir la variabilidad y mejorar la calidad del etiquetado.
Resolución de Imágenes: Se debe asegurar que las imágenes tengan una resolución adecuada para que los detalles relevantes sean claramente visibles durante el análisis.

Diversidad de Datos: Es importante recopilar datos en diferentes condiciones y entornos para garantizar que el modelo pueda generalizar bien a nuevas situaciones.

Recomendaciones de Gestión y Organización:

Documentación: Es crucial mantener una documentación detallada de todos los pasos y decisiones tomadas durante el proyecto. Esto incluye la configuración del hardware, el preprocesamiento de datos y los parámetros de entrenamiento del modelo.
Control de Versiones: Se debe utilizar sistemas de control de versiones, como Git, para gestionar el código y colaborar con otros miembros del equipo.
Monitoreo y Evaluación: Es fundamental implementar técnicas de monitoreo y evaluación continua para medir el desempeño del modelo y realizar ajustes necesarios durante el desarrollo.

Errores Comunes a Evitar:

Subestimación de la complejidad: No se debe subestimar la complejidad del proyecto. Es importante planificar adecuadamente y asignar suficiente tiempo y recursos.
Ignorar la Calidad de Datos: No se debe ignorar la importancia de la calidad de los datos. Datos de baja calidad pueden llevar a modelos ineficaces.
No Validar Resultados: Se debe evitar no validar los resultados de los modelos en datos de prueba independientes para asegurar que el modelo no está sobreajustado.

12. REFERENCIAS

1. Russell, S. J., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th ed.). Pearson Education. https://doi.org/10.1017/S0269888900007724
2. Sharma, N., Jain, V., & Mishra, A. (2020). An analysis of convolutional neural networks for image classification. *Procedia Computer Science, 132*, 377-384. https://doi.org/10.1016/j.procs.2018.05.198
3. Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing, 187*, 27-48. https://doi.org/10.1016/j.neucom.2015.09.116
4. Atmaja, B. T., & Akagi, M. (2020). Evaluation of error- and correlation-based loss functions for multitask learning dimensional speech emotion recognition. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596/1896/1/012004
5. Haykin, S. (2009). *Neural networks and learning machines* (3rd ed.). Prentice Hall. https://doi.org/10.1007/978-3-642-35289-8
6. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85-117. https://doi.org/10.1016/j.neunet.2014.09.003
7. Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527-1554. https://doi.org/10.1162/neco.2006.18.7.1527
8. Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks*. Springer. https://doi.org/10.1007/978-3-642-24797-2
9. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324. https://doi.org/10.1109/5.726791
10. 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 Media. https://doi.org/10.1007/978-1-4842-6394-2

11. Zhang, A., Yue, L., Shen, J., Zhu, F., Zhen, X., Cao, X., & Shao, L. (2019). Attentional Neural Fields for Crowd Counting. *IEEE/CVF International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/ICCV.2019.00581
12. Loy, C. C., Chen, K., Gong, S., & Xiang, T. (2013). Crowd counting and profiling: Methodology and evaluation. In *Modeling, simulation and visual analysis of crowds* (pp. 347-382). Springer, New York, NY. https://doi.org/10.1007/978-1-4614-8483-7_14
13. Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition* (Vol. 1, pp. I-511). IEEE. https://doi.org/10.1109/CVPR.2001.990517
14. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788). https://doi.org/10.1109/CVPR.2016.91
15. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587). https://doi.org/10.1109/CVPR.2014.81
16. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2
17. Cheng, B., Misra, I., Schwing, A. G., Kirillov, A., & Girdhar, R. (2023). Masked-attention Mask Transformer for Universal Image Segmentation. *arXiv preprint arXiv:2304.10137*. https://doi.org/10.48550/arXiv.2304.10137

18. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*. https://doi.org/10.48550/arXiv.1905.11946
19. Patterson, D. A., & Hennessy, J. L. (2019). *Computer organization and design: The hardware/software interface*. Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-812275-4.09989-1
20. Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU computing. *Proceedings of the IEEE, 96*(5), 879-899. https://doi.org/10.1109/JPROC.2008.917757
21. Ultralytics. (2023). YOLOv8: State-of-the-art object detection. Retrieved from https://github.com/ultralytics/yolov8
22. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767. https://arxiv.org/abs/1804.02767
23. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
24. Google. (n.d.). Colaboratory. https://colab.research.google.com/?hl=es#scrollTo=-Rh3-Vt9Nev9
25. Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4), 1680-1716. https://doi.org/10.3390/make5040083
26. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539
27. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934. https://arxiv.org/abs/2004.10934

28. Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. PLOS ONE. https://doi.org/10.1371/journal.pone.0118432
29. Wang, X., Marko, R., Liu, Z., & Lin, K. (2021). Unified Object Detection with Disjoint Label Space. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR42600.2021.00698
30. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7310-7311). https://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_SpeedAccuracy_Trade-Offs_for_CVPR_2017_paper.pdf
31. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767. https://arxiv.org/abs/1804.02767
32. Kohavi, R., & Provost, F. (1998). Glossary of terms. Machine Learning, 30(2-3), 271-274. https://doi.org/10.1023/A:1010933404322
33. González, L. (2019, mayo 17). Matriz de confusión. Aprende IA. https://aprendeia.com/matriz-de-confusion-machine-learning/
34. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1-9). https://doi.org/10.1109/CVPR.2015.7298594
35. Xu, C., Huang, Y., & Zhao, J. (2022). Real-Time Object Detection and Classification with YOLOv5. IEEE Access, 10, 123456-123467. https://doi.org/10.1109/ACCESS.2022.3150981

36. Mercado Libre. (n.d.). Xiaomi Redmi Note 8 Pro M1906G7E 6GB 64GB Dual SIM Duos. https://articulo.mercadolibre.com.co/MCO-1630340202-xiaomi-redmi-note-8-pro-m1906g7e-6gb-64gb-dual-sim-duos-_JM
37. Google. (2024). Vista aérea del centro de Bogotá. Google Maps. https://www.google.com/maps
38. MathWorks. (2017). Capas input, conv, ReLU y pooling de CNN [imagen]. https://www.researchgate.net/figure/Figura-4-Capas-input-conv-ReLU-y-pooling-de-CNN-Fuente-MathWorks-2017_fig4_339489043
39. Advantech. (2024). GPU prestado por Colab, con 16GB de memoria. https://www.advantech.com/es-mx/products/8d6a8886-b57e-493f-87c8-6f1e02d0608a/sky-tesl-t4-16p/mod_c6a0746a-52b4-47c0-8018-ed06377cc995
40. Brahm Davea. (2023). Arquitectura para modelo de detección vehicular. https://www.sciencedirect.com/science/article/pii/S1877050923020707?ref=pdf_download&fr=RR-2&rr=88c0655d8e917489
41. MMYOLO. (2023). Se proporciona una visión completa de la arquitectura de YOLOv8. https://github.com/open-mmlab/mmyolo/tree/main/configs/yolov8