ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

UNIDAD DE TITULACIÓN

ANÁLISIS DE RENDIMIENTO DE YOU ONLY LOOK ONCE, RETINANET Y SINGLE SHOT DETECTOR APLICADO A LA DETECCIÓN Y CONTEO VEHICULAR

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN SISTEMAS DE INFORMACIÓN COM MENCIÓN EN INTELIGENCIA DE NEGOCIOS Y ANALÍTICA DE DATOS MASIVOS

IVÁN ANDRÉS BUITRÓN TANDALIA
andresbuitront@gmail.com

Director: Sang Guun Yoo Park
sang.yoo@epn.edu.ec

2023

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación “ANÁLISIS DE RENDIMIENTO DE YOU ONLY LOOK ONCE, RETINANET Y SINGLE SHOT DETECTOR APLICADO A LA DETECCIÓN Y CONTEO VEHICULAR” desarrollado por Iván Andrés Buitrón Tandalia, estudiante de la Maestría en Sistemas de Información con mención en Inteligencia de Negocios y Analítica de Datos Masivos, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

Sang Guun Yoo Park

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, Iván Andrés Buitrón Tandalia, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Firmado electrónicamente por:
IVAN ANDRES BUITRON TANDALIA

Iván Andrés Buitrón Tandalia

DEDICATORIA

Primero a Dios y a mi Madre Dolorosa, pilares fundamentales espirituales en mi vida, que me llenan de bendiciones diarias y me permiten ser más para servir mejor…

A mi padre, que su resiliencia me sigue dando lecciones de cómo superar la vida día a día con amor; A mi madre, mi mayor inspiración, mi confidente y mi apoyo incondicional, siempre lo que hace, lo hace con amor, todo lo que fui, soy y seré es por y para ti; A mi hermana, tu ejemplo es la mayor muestra de superación y lucha para cumplir mis sueños.

A mis padrinos, abuelitos, tíos, primos, amigos, quienes son apoyo constante en mi vida en cada paso, les agradezco por creer en mí incluso cuando yo dudaba de mí mismo. Mi éxito es vuestro éxito, y cada paso que he dado ha sido posible gracias a todos ustedes.

Esta tesis no solo es un logro personal, sino también un testimonio de vuestro amor, dedicación y sacrificio. Cada página escrita es un tributo a su presencia en mi vida y a la influencia positiva que han tenido en mi desarrollo académico y personal.

AGRADECIMIENTO

A la prestigiosa Universidad Politécnica Nacional, por haberme otorgado la invaluable oportunidad de recibir una beca para continuar mis estudios apostando por mi talento con convicción y excelencia académica.

Agradezco a las autoridades de la universidad por su visión y liderazgo al establecer oportunidades de crecimiento a los estudiantes. Su compromiso con la educación y la igualdad de oportunidades ha tenido un impacto profundo en mi vida y en la de muchos otros estudiantes.

Al doctor Sang Yoo, por su indispensable guía, apoyo y dedicación como mi tutor de tesis. A lo largo de este proceso, su liderazgo y experiencia han sido fundamentales para el éxito de mi trabajo de investigación y para mi desarrollo académico y profesional.

A la doctora María Pérez, coordinadora del programa, que con su compromiso y profesionalismo ha sabido comprender las necesidades de sus maestrantes brindando orientación y una guía valiosa.

A los profesores, que cada uno de ustedes ha sido una fuente inagotable de inspiración y motivación, por su disposición para compartir sus conocimientos y experiencias con técnicas que despiertan interés y pasión por los estudiantes.

ÍNDICE DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

La inteligencia artificial ha tenido un crecimiento significativo en las últimas décadas, debido a esto se han desarrollado varias arquitecturas para la detección, clasificación y reconocimiento de objetos, actualmente hay varias alternativas que cumplen con esta finalidad sin embargo no existe un estructura rígida de cómo se forman estas arquitecturas, este estudio presenta el análisis actualizado de rendimiento en la detección y conteo vehicular utilizando You Only Look Once (YOLOv8) versión 8, RetinaNet (RN) y Single Shot Detector (SSD) en tiempo real, usando como entorno de reentrenamiento Google Colaboratory y un conjunto de imágenes personalizado. Se utilizó la metodología de Investigación-Acción para desarrollar el caso práctico y se hizo una revisión sistemática de literatura para determinar el estado del arte de esta problemática. Para extraer características, se usó RESNET-50 y Mobilnet en RN y SSD respectivamente teniendo como resultado que YOLOv8 (mismo que presenta más ajustes desde su creación) presenta el mejor desempeño en función del tiempo de detección, precisión tomando en cuenta los fotogramas que se requiere ser analizados para que tenga un uso en tiempo real y facilidad de implementación.

Palabras clave: Vehicle Detection, Vehicle Counting, YOLOv8, SSD, RetinaNet, Computer Vision, Artificial Intelligence.

ABSTRACT

Artificial intelligence has experienced significant growth in recent decades. As a result, several architectures have been developed for object detection, classification, and recognition. Currently, there are several alternatives that fulfill these purposes; however, there is no rigid framework defining how these architectures are formed. This work presents an updated performance analysis of real-time vehicle detection and counting using You Only Look Once (YOLOv8) version 8, RetinaNet (RN), and Single Shot Detector (SSD). For such analysis, the Google Colaboratory was used as the main re-training environment. The Research-Action methodology was employed to develop the practical case, and a systematic literature review was also con-ducted to determine the state of the art in this problem domain. For feature extraction, RESNET-50 and MobileNet were used in RN and SSD, respectively. The results indicated that YOLOv8 (which has undergone the most adjustments since its inception) exhibits the best performance in terms of detection time, precision considering the frames that need to be analyzed to enable real-time usage, and ease of implementation.

Keywords: Vehicle Detection, Vehicle Counting, YOLOv8, SSD, RetinaNet, Computer Vision, Artificial Intelligence.

1. INTRODUCCIÓN

Han pasado alrededor de sesenta años desde las primeras publicaciones sobre la visión por computadora. (Roberts, 1963), publicó su tesis de doctorado en el MIT llamado “Machine perception of three-dimensional solids”, donde discute las posibilidades de extraer información de figuras geométrica en tres dimensiones vistas en perspectivas de dos dimensiones de bloques. Otro acontecimiento que se puede indicar es el trabajo publicado por (Papert, 1966) donde se realiza un experimento para describir imágenes por computador; experimento que se realizó en MIT. Estos son los primeros acercamientos documentados y empíricos dentro del desarrollo de reconocimiento de patrones.

Desde entonces, las estrategias de visión por computadora e inteligencia artificial en general han ido aplicándose en varias soluciones de la vida cotidiana y, a medida que la potencia de procesamiento en las nuevas computadoras aumentaba (Sze et al., 2017) y se disponía de más datos e imágenes digitales, la visión por computadora comenzó a avanzar rápidamente. Alrededor de la década de 1980, se desarrollaron algoritmos más sofisticados para realizar tareas como el reconocimiento de patrones y la detección de objetos en imágenes. Uno de los hitos importantes en este campo fue la creación del sistema "Neocognitron" por Kunihiko (Fukushima, 1980), trabajo que se considera como uno de los primeros modelos de redes neuronales convolucionales, una técnica fundamental en la visión por computadora actual.

En la década de los 90s, el interés en la visión por computadora se intensificó aún más con la aplicación de técnicas de aprendizaje automático, como las redes neuronales artificiales. El enfoque en esta época se centraba en el reconocimiento de objetos y rostros, y se lograron avances significativos en esta área. Sin embargo, la visión por computadora todavía enfrentaba desafíos importantes, como la variabilidad en la iluminación (Bahri et al., 2020), la oclusión de objetos y la detección de objetos en escenas complejas.

En los últimos 20 años, la visión por computadora ha experimentado un rápido progreso gracias a los avances en el aprendizaje profundo y las redes neuronales convolucionales. Estas técnicas han permitido mejoras significativas en áreas como el reconocimiento de objetos, el seguimiento de objetos en videos, la detección facial y el análisis de imágenes médicas. Además, la disponibilidad de grandes conjuntos de datos etiquetados, como ImageNet (Deng et al., 2009), ha impulsado aún más el desarrollo de algoritmos de visión

por computadora y se pueden dividir históricamente en (1) "período de detección de objetos tradicional (hasta antes de 2014)" y (2) "período de detección basado en aprendizaje profundo (después de 2014)", como se muestra en la Figura 1 (Zou et al., 2019).

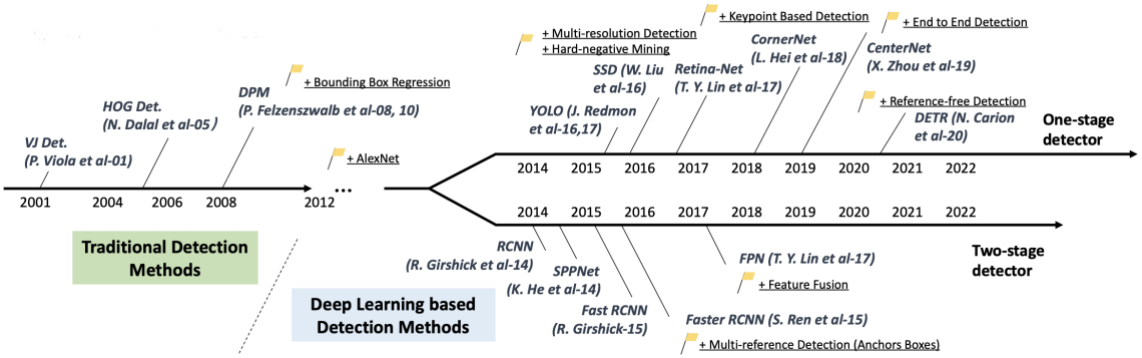

Figura 1 - Línea de tiempo basado en métodos de detección tradicionales y de aprendizaje profundo (cita)

En el artículo titulado “Object Detection in 20 Years: A Survey” (Zou et al., 2019) resume estos avances, donde destacan el detector Viola-Jones y el detector HOG, que lograron la detección de rostros en tiempo real y mejoraron la detección considerando características de escala. En este período, se presentan los detectores de dos etapas como R-CNN, SPPNet, Fast R-CNN y Faster R-CNN, que mejoraron la velocidad y precisión mediante características espaciales y propuestas de regiones.

Cuando la necesidad migró hacia aplicaciones que funcionen en tiempo real, se presentaron algunos inconvenientes con detectores de dos etapas debido al coste computacional y a la propia estrategia que siguen para detectar objetos, esto género que surjan nuevas propuestas paralelas enfocadas en la rapidez y eficiencia que hizo ganar mayor popularidad a YOLOv8, RN y SSD, el principal cambio era que realizan la detección en una sola pasada (H. Zhang & Cloutier, 2022), lo que los hace efectivos para procesar escenas en tiempo real y con altos fotogramas por segundo y estas tres arquitecturas en particular destacan por la cantidad de documentación y análisis que existen.

En relación con eso, este trabajo de investigación busca comparar el rendimiento de los algoritmos YOLO, RN y SSD, en sus versiones más recientes disponibles, realizando un Fine Tunning (Dodge et al., 2020) o custom training con nuestro propio set de datos, para

realizar la detección de automotores livianos y pesados encapsulados en nuestra clase “vehículo” y, posteriormente, contarlos en tiempo real.

El resto del documento se encuentra estructurado de la siguiente manera. La sección 2 revisa el estado del arte de las arquitecturas. En la sección 3, explica la metodología utilizada en este trabajo, las herramientas que fueron utilizadas en el experimento y los detalles de cada implementación, para desarrollar la detección y conteo de vehículos. En la sección 4 se presenta una pequeña discusión y finalmente, la Sección 5 concluye el presente trabajo.

1.2. Pregunta de investigación

¿Qué arquitectura presenta el mejor el rendimiento comparativo de los algoritmos YOLO, SSD y RetinaNet en la detección y conteo de vehículos?

1.3. Objetivo general

Analizar el rendimiento de los métodos de detección de objetos You Only Look Once (YOLO), RetinaNet (RN) y Single Shot Detector (SSD) para la detección y conteo de vehículos.

1.4. Objetivos específicos

- Comprender los conocimientos que están detrás de la implementación de YOLO, RN y SSD.
- Realizar un ajuste fino sobre las tres arquitecturas para la detección y conteo de vehículos con nuestro propio conjunto de datos.
- Comparar las herramientas YOLO, RN y SSD basados en escenarios similares.

1.5. Marco Teórico

Redes neuronales convolucionales

Las redes neuronales convolucionales son un tipo especializado de red neuronal artificial (ANN, por sus siglas en inglés) que se utilizan para el análisis de imágenes. Dado que las computadoras ven la imagen como una matriz de números que representan cada píxel, es importante que la relación entre los pixeles (valores de niveles de gris), permanezca incluso

después de que la imagen se procese a través de la red. Para salvar esta relación espacial entre píxeles, se utilizan redes neuronales de convolución que tienen diferentes operaciones matemáticas apiladas una encima de la otra para crear capas de la red (Zhao et al., 2019).

Las convoluciones son operaciones matemáticas que toman dos funciones y genera como resultado una tercera, durante este procedimiento se aplican técnicas que superponen una función sobre la otra para producir la tercera. Dentro del procesamiento de imágenes, se añaden pixeles dentro de la imagen, y esto da como resultado una mejor detección de características en la imagen (Bhatt et al., 2021).

La columna vertebral de estos modelos de visión por computadora se apoya en el concepto de redes neuronales convolucionales (CNN) (Zhao et al., 2019) que desempeñan un papel fundamental porque están desarrollados específicamente para analizar imágenes y extraer características relevantes de ellas.

Aunque los conceptos y las primeras investigaciones en redes neuronales convolucionales se remontan a la década de 1980, su popularidad y aplicación generalizada se han producido en los últimos años con los avances en el aprendizaje profundo y la disponibilidad de grandes conjuntos de datos etiquetados.

Ergo el aprendizaje profundo, ha impulsado el campo de la visión por computadora al permitir la extracción automática de características, mejorar la precisión en la clasificación de imágenes, impulsar la detección y localización de objetos en tiempo real, mejorar la segmentación semántica y lograr avances significativos en el reconocimiento facial y de emociones. Estos avances han abierto nuevas posibilidades en aplicaciones como la visión artificial, la conducción autónoma, la vigilancia de video, la medicina y muchas más.

Intersección sobre unión

La Intersección sobre unión, también conocida como Intersection over Union (IoU) en inglés, es una métrica comúnmente utilizada en algoritmos de detección de objetos de una etapa, como el detector de objetos basado en Redes Neuronales Convolucionales (CNN) conocido como YOLO.

La IoU se utiliza para evaluar la precisión de la detección de objetos al comparar la superposición entre la detección generada por el modelo y la ubicación real del objeto en

la imagen (Mahdi et al., 2020). La IoU se calcula dividiendo el área de la intersección entre la detección y el objeto real por el área de la unión entre ambos. La fórmula matemática para calcular la IoU es la siguiente:

IoU = (Área de Intersección) / (Área de Unión)

Un valor alto de IoU (cerca de 1) indica una alta superposición y, por lo tanto, una detección precisa del objeto. Por el contrario, un valor bajo de IoU (cerca de 0) indica que la detección no se superpone mucho con el objeto real, lo que sugiere una baja precisión como se visualiza en la figura 2.

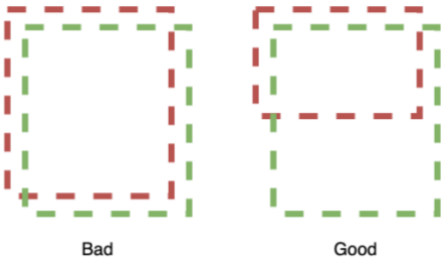

Figura 2 - Representación de un ejemplo de la intersección sobre la unión mala y buena

En el contexto de los algoritmos de una etapa, como YOLO, la IoU se utiliza como un umbral para filtrar las detecciones. Solo se consideran como detecciones válidas aquellas que tienen un IoU por encima de un valor de umbral predefinido. Esto ayuda a eliminar detecciones falsas o con una baja precisión como se puede ver en la figura 3.

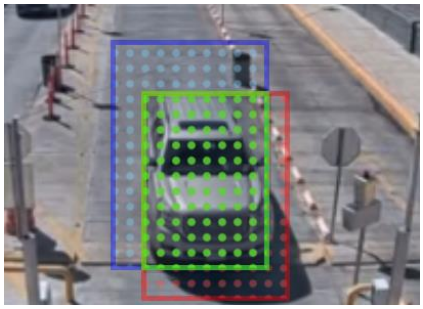

Figura 3 - Representación de la aplicación de intersección sobre unión aplicada en las arquitecturas de una etapa

SSD: Single Shot MultiBox Detector

La primera versión de SSD, fue en 2015/2016, dentro del artículo "SSD: Single Shot MultiBox Detector" desarrollado por Wei (Liu et al., 2015), que introduce un método de detección de objetos denominado de esta manera. El enfoque de SSD aborda el desafío de detectar objetos en imágenes de manera rápida y precisa de acuerdo con los autores, se presenta la arquitectura de SSD, que propone una solución más eficiente y efectiva.

SSD utiliza una red neuronal convolucional (CNN) para realizar la detección de objetos en una sola pasada ("single shot"). A diferencia de otros métodos que utilizan una búsqueda exhaustiva de regiones de interés, SSD propone la predicción simultánea de múltiples cuadros delimitadores ("bounding boxes") y las clases de objetos correspondientes en diferentes escalas y niveles de resolución.

Este método, se diferencia debido al espacio de salida de los cuadros delimitadores en un conjunto de cuadros predeterminados sobre diferentes relaciones de aspecto y escalas por ubicación del mapa de características.

El enfoque SSD se basa en una red convolucional de avance que produce una colección de cuadros delimitadores de tamaño fijo y puntuaciones para la presencia de instancias de clase de objeto en esos cuadros, seguido de un paso de supresión no máxima para producir las detecciones finales.

Mapas de características según (Gómez-Huélamo et al., 2022) que son multiescala para la detección. Agregamos capas de características convolucionales al final de la red base truncada. Estas capas disminuyen de tamaño progresivamente y permiten predicciones de detecciones a múltiples escalas.

Dentro del experimento (Tan et al., 2021) de SSD, originalmente se utiliza una red de extracción de características de tipo VGG-16, sin embargo, existe un apartado del texto donde indican que puede funcionar bien con cualquier red adyacente para realizar esta labor, por lo que, para aportar un grado más significativo de investigación, en este trabajo, se usa la red MobileNet.

MobileNet

El enfoque principal de MobileNet es lograr una eficiencia computacional y de tamaño de modelo reducido sin comprometer demasiado la precisión en la tarea de clasificación de imágenes.

El funcionamiento de MobileNet se basa en el uso de operaciones de convolución separable en lugar de convoluciones estándar. La convolución separable se divide en dos etapas: una convolución de punto y una convolución de canal. En la convolución de punto, se realizan operaciones de convolución 1x1 para reducir la dimensión del espacio de características según (Howard et al., 2017) como se muestra en la figura 4.

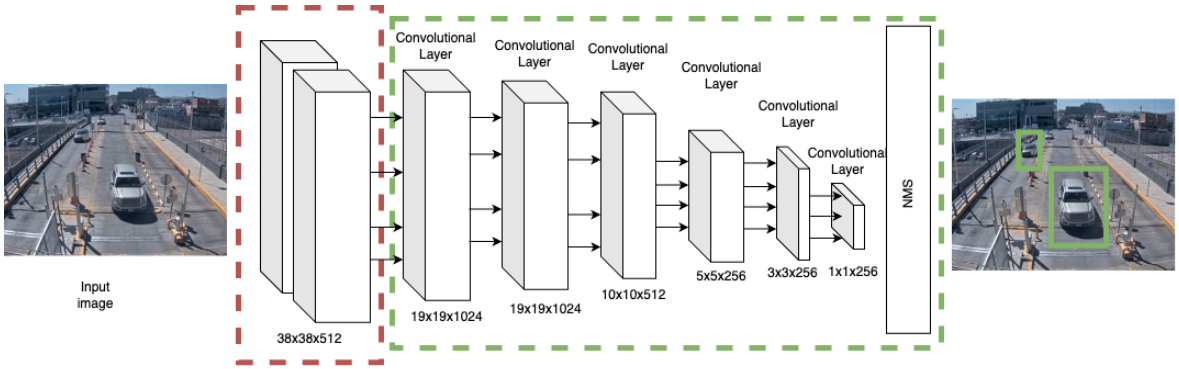

Figura 4 - Arquitectura de Red Neuronal Single Shot Detector con MobileNet

Pirámide de características

El artículo "Feature Pyramid Networks for Object Detection" (T.-Y. Lin et al., 2016) introduce una arquitectura de red neuronal convolucional llamada Feature Pyramid Network (FPN) para abordar el desafío de detectar objetos en imágenes de diferentes escalas.

El FPN aborda la limitación de las redes convolucionales estándar que tienen dificultades para detectar objetos pequeños debido a las capas de pooling como menciona (He et al., 2014) que reducen la resolución espacial. La arquitectura FPN propone una pirámide de características, donde se construyen mapas de características en diferentes escalas, capturando tanto las características finas como las características contextuales de la imagen.

YOLO: You Only Look Once

YOLO, cuenta con 16 versiones desde sus inicios en 2016 (Terven & Cordova-Esparaza, 2023), las primeras versiones de YOLO trabajaban con PASCAL VOC y a partir de YOLOv3 usa COCO (Redmon et al., 2016) y su premisa fundamental consiste en ser un método de detección de objetos en tiempo real que utiliza una única pasada a través de una red neuronal convolucional. En lugar de dividir la imagen en regiones de interés, YOLO divide la imagen en una cuadrícula y asigna a cada celda la responsabilidad de detectar objetos. Utiliza múltiples capas de convolución para predecir cuadros delimitadores y probabilidades de clases de objetos en cada celda. Aunque la versión original tenía limitaciones en la detección de objetos pequeños, versiones posteriores como YOLOv2 (Redmon & Farhadi, 2016) y YOLOv3 (Redmon & Farhadi, 2018) mejoraron la precisión y resolvieron algunos problemas.

Para YOLOv4 se cambió el ecosistema de darknet a pytorch y agregaron características de ampliación y reducción de escala (Bochkovskiy et al., 2020); en YOLOv5 (Jocher, 2020) o YOLOX existieron cambios en la forma de la estructura, agregaron una cabeza desacoplada y sustituyeron cambios en las capas de detección; para YOLOv7 se mejoró en la precisión del modelo que finalmente en 2023, YOLOv8 (Jocher et al., 2023) utiliza un modelo sin anclaje con un cabezal desacoplado para procesar de forma independiente la objetualidad, la clasificación y tareas de regresión.

Desde YOLOv5, todos los modelos oficiales de YOLO han perfeccionado el equilibrio entre velocidad y precisión, ofreciendo diferentes escalas modelo para adaptarse a aplicaciones específicas y requisitos de hardware. Por ejemplo, estas versiones a menudo proporcionan modelos livianos optimizados para dispositivos de borde, precisión comercial para reducir la complejidad computacional y tiempos de procesamiento más rápidos.

Las variantes de YOLO como YOLO-S, YOLO-M y YOLO-XL representan diferentes versiones o arquitecturas del modelo original con ciertos cambios y mejoras.

En general, las diferencias entre las variantes de YOLO se centran en aspectos como el tamaño y profundidad de la red neuronal convolucional utilizada, el número de capas y filtros, y la precisión y velocidad de detección definido en la tabla 1.

Tabla 1 - Rendimiento y Características de las redes de YOLOv8.

Modelo	Tamaño (pixeles)	mAP	Velocidad CPU ONNX	Velocidad A100 TensorRT	Parámetros (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	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

La Arquitectura de YOLOv8 tiene una etapa de extracción de características, se utilizan múltiples capas convolucionales para procesar la imagen de entrada y extraer características de alto nivel. Estas capas convolucionales se encargan de detectar patrones y características relevantes en la imagen, como bordes, texturas y formas.

Luego, se realiza una reducción de dimensionalidad mediante capas de agrupación (pooling), que reducen el tamaño de las características mientras se conserva su información más relevante. Esta reducción de dimensionalidad permite obtener una representación más compacta de la imagen y facilita la detección de objetos tal y como se muestra en la figura 5.

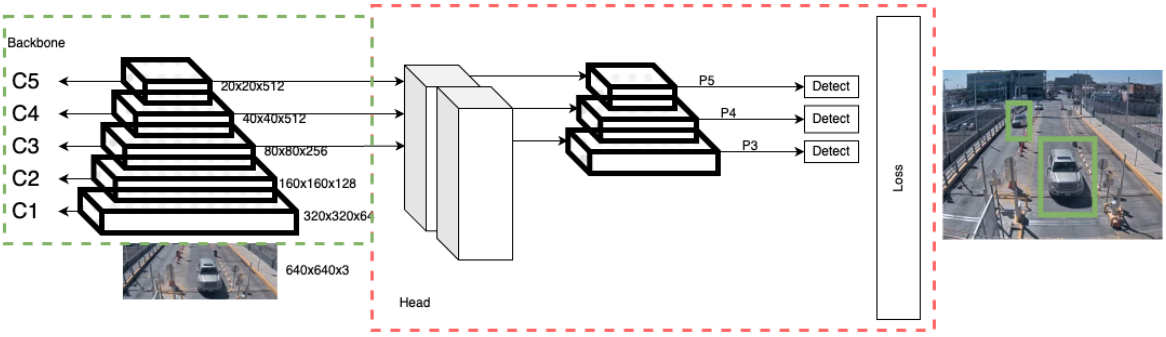

Figura 5 - Arquitectura de Red Neuronal YOLO

RN: RetinaNet

Otro de los métodos que están puestos a prueba en este trabajo es Retinanet; RetinaNet es una arquitectura de red neuronal convolucional (CNN) propuesta por (T.-Y. Lin et al., 2018) para la tarea de detección de objetos en imágenes. La característica

principal de RetinaNet es la combinación de dos componentes clave: la Red de Cabeza de Detección (Detection Head) y las Anclas de Retina (Retina Anchors).

La Red de Cabeza de Detección es una subred que toma las características de entrada y produce predicciones tanto para la clasificación de objetos como para la regresión de cuadros delimitadores. Utiliza una combinación de capas convolucionales y capas de convolución en forma de cascada (convolución 3x3 y convolución 1x1) para extraer características relevantes y realizar las predicciones como menciona (L. Zhang et al., 2021).

Las Anclas de Retina son una técnica que se utiliza para generar regiones de interés en diferentes escalas y aspectos de la imagen. Estas anclas actúan como cajas delimitadoras predefinidas que cubren diferentes áreas de la imagen. La red aprende a clasificar y ajustar estas anclas para detectar objetos en diferentes tamaños y formas.

RetinaNet también utiliza una pérdida focal (focal loss) como función de pérdida durante el entrenamiento. Esta función de pérdida se enfoca en las regiones de difícil clasificación y evita que los objetos de fondo abrumen las predicciones, lo que mejora el equilibrio entre las clases según (T.-Y. Lin et al., 2018).

ResNet50

ResNet-50 es una arquitectura de red neuronal convolucional (CNN) profunda que fue propuesta por (He et al., 2015; Wightman et al., 2021). "ResNet" es una abreviatura de "Residual Network" (Red Residual), y el número "50" se refiere al número total de capas en la red. La característica distintiva de ResNet-50 es su utilización de conexiones residuales, que permiten que la información salte una o más capas en lugar de fluir de manera secuencial a través de todas las capas. Estas conexiones residuales ayudan a abordar el problema del desvanecimiento del gradiente y facilitan el entrenamiento de redes neuronales más profundas.

En la figura 6 se muestra una representación de la solución de ResNet50 como extractor de características y RetinaNet como detector de objetos.

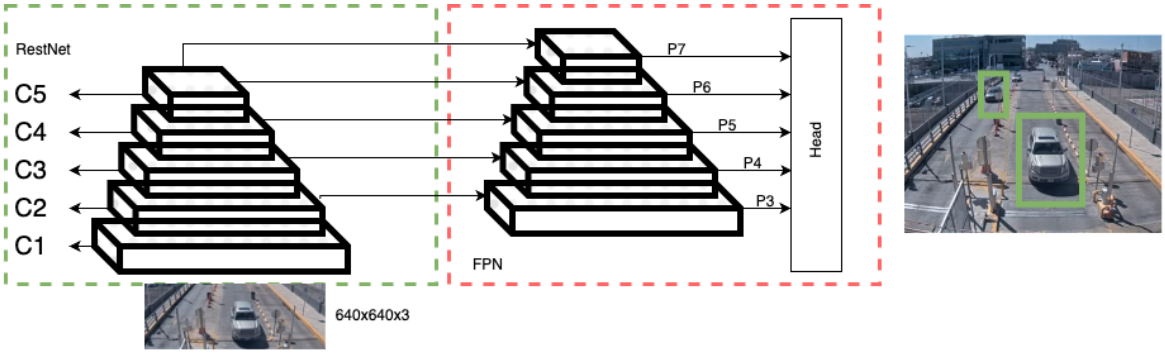

Figura 6 - Arquitectura de Solución basada en ResNet-50 y RetinaNet

2. METODOLOGÍA

Según (Altrichter et al., 2002), en su obra Educación, Conocimiento e Investigación Acción, la investigación acción tiene como propósito crear sociedades de personas comprometidas con aprender en relación directa con las circunstancias del medio en el que habita, de este modo los problemas sociales que se presentan en las comunidades deberían servir como impulso para realizar investigaciones públicas que brinden soluciones para el colectivo.
El modelo de Lewis reconoce estas necesidades que se habían presentado anteriormente en los trabajos de Kemmis y muestra como alternativa el modelo cíclico que presenta un flujo constante de mejora continua que se presenta en la figura 7.

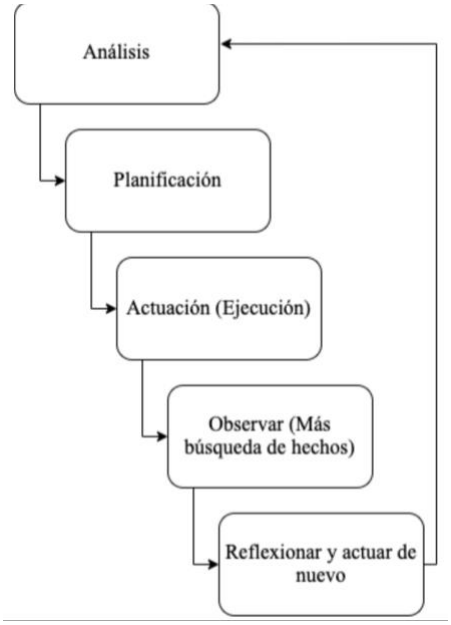

Figura 7 - Etapas de la metodología de Investigación-Acción

Análisis, búsqueda de hechos y reconceptualización: Durante esta etapa del desarrollo de este trabajo, se pretende indagar los conceptos asociados a visión por computadora, YOLO, SSD, RN, y adicionalmente, recopilar los clips que van a servir para entrenar estos algoritmos.

En primer lugar, se va a identificar claramente el objetivo de la investigación, que es comparar el desempeño de los algoritmos en la detección de objetos. Luego, realizar una revisión bibliográfica exhaustiva para comprender los principios y características de cada algoritmo. A continuación, diseñar un experimento que incluya la implementación de los algoritmos y la recopilación de datos utilizando un conjunto de prueba adecuado. Analizar los resultados obtenidos, comparando métricas de evaluación predefinidas. Finalmente, discutir las conclusiones obtenidas, recomendar el algoritmo más adecuado para diferentes escenarios y proponer áreas de investigación futura.

2.1. Análisis

Dentro de la Etapa de Análisis, se realizó una revisión de literatura de trabajos existentes con la finalidad de entender su estado del arte. Para esta parte del trabajo, se realizó un proceso riguroso y estructurado que tiene como objetivo recopilar, evaluar y analizar exhaustivamente las bibliotecas digitales y/o repositorios de científicos en búsqueda de evidencia científica disponible sobre YOLO, SSD, RN. Usamos el enfoque de tres etapas para realizar esta tarea tal y como se muestra en la figura 8, cada etapa tiene subtareas que permiten llevar este proceso de una manera organizada y objetiva, que elimine cualquier sesgo que pueda existir.

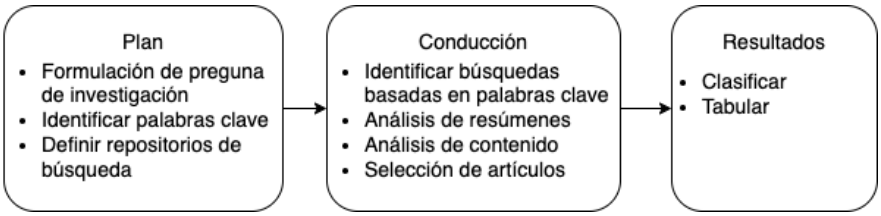

Figura 8 - Etapas de la estrategia adoptada basada en la revisión sistemática

En la etapa de análisis y planificación de una revisión sistemática de literatura, se estableció claramente la pregunta de investigación y se definieron los objetivos y los criterios de inclusión y exclusión de los estudios. Además, se identificaron las palabras clave relevantes y se elaboró una lista de términos que serán utilizados en la búsqueda bibliográfica.

Asimismo, se determinaron los repositorios o bases de datos electrónicas donde se llevó a cabo la búsqueda como menciona (Williams et al., 2021).

Basado en el esquema antes mencionado, en primer lugar, se plantearon las siguientes preguntas de investigación:
¿Hay soluciones que comparen YOLO, SSD, RN?
¿Qué Arquitecturas de Redes Neuronales son usadas con mayor frecuencia en la detección en tiempo real?
¿Qué Arquitecturas de Redes Neuronales son usadas con mayor frecuencia para el conteo vehicular?

Estas preguntas se encuentran relacionadas estrechamente con nuestro objetivo de investigación, que es detectar y contar vehículos con los modelos YOLO, SSD y RN.

Luego se procedió a identificar las palabras claves para contestar las preguntas generadas, las cuales fueron: You Only Look Once, Single Shot Detector, RetinaNet, visión por computadora, aprendizaje profundo, detección y clasificación, redes neuronales, tiempo real.

Finalmente, para terminar con esta fase inicial, se seleccionaron los lugares donde íbamos a realizar las búsquedas, Las bases de datos de investigación son tomadas en cuenta en función de cadenas de búsqueda especificadas para búsquedas de artículos. Seleccionado las bases de datos fueron ACM Digital Library, IEEE Xplore, Arxiv, Google Scholar, Scopus.

En el proceso de conducción de una revisión sistemática de literatura, se llevó a cabo la búsqueda bibliográfica utilizando las palabras clave previamente identificadas. Se aplicaron filtros y criterios de búsqueda para obtener resultados más relevantes. Posteriormente, se revisaron los resultados obtenidos y se identificaron los estudios potencialmente relevantes en base a los títulos y resúmenes de los artículos, descartando aquellos que no cumplen con los criterios de inclusión establecidos. Luego, se realizó un análisis más detallado de los resúmenes de los artículos seleccionados, evaluando su relevancia y pertinencia para la pregunta de investigación. Los artículos seleccionados se sometieron a un análisis exhaustivo, donde se leyó en su totalidad y se extrajeron los datos relevantes utilizando herramientas como matrices de extracción de datos. Finalmente, se realizó una selección final de los artículos incluidos en la revisión, considerando su relevancia, calidad metodológica y contribución a la pregunta de investigación planteada.

Una vez filtrados los artículos, éstas fueron clasificados y tabulados de manera sistemática. En la Tabla 2, se resumen los resultados de los artículos que se encontraron, donde se tuvo en cuenta, el título y la fecha de publicación, las referencias de código fuente de la implementación, las webs oficiales de los creadores de las arquitecturas, el resumen y el contenido de cada artículo.

Tabla 2 - Tabla de resultados de la revisión sistemática

	ACM Digital Library	**IEEE Xplore**	**Arxiv**	**Google Scholar**	**Scopus**	**Parcial**	**Filtro Título**	**Filtro de Resumen**	**Filtro de Contenido**	**Final**
SSD	86	24	11	6	29	156	100	32	14	10
YOLO	101	14	32	19	15	181	87	39	43	12
RN	57	19	4	8	2	90	45	17	20	8
Deep Learning Y SSD, YOLO, RN	14	8	4	2	0	28	11	1	10	6
Car Detection Y SSD, YOLO, RN	16	6	4	2	1	29	19	1	6	3
Car Counting Y SSD, YOLO, RN	11	6	66	2	14	99	90	5	2	2
Others	8	3	6	11	4	32	12	10	6	4

Con esto, en conclusión, se obtiene una visión integral y actualizada del estado del conocimiento en el área de arquitecturas de una sola etapa en detección de objetos. Esto permite identificar a YOLOv8 como un método que ha sido bastante estudiado a lo largo del tiempo, destacar tendencias y obtener evidencia sólida para respaldar futuros proyectos o en esta área.

A continuación, se muestra el análisis de algunos de los trabajos previos que fueron seleccionados, para entender sus funcionalidades y en caso de existir, sus limitaciones.

(Benjdira et al., 2018) en su artículo "Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3" se centra en comparar las arquitecturas de Faster R-CNN y YOLOv3 en la detección de automóviles utilizando imágenes capturadas desde UAV. El artículo evalúa y compara el rendimiento de ambas arquitecturas en términos de precisión y velocidad, teniendo en cuenta las características y desafíos específicos de la detección de objetos en imágenes aéreas, las limitaciones implicadas están relacionadas con YOLO, puesto que actualmente se ha actualizado a la versión 8 que presenta cambios significativos desde la 3 que impactan directamente en el desempeño.

En resumen, el artículo "A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO" de (C.-J. Lin et al., 2021) describe un sistema en tiempo real para contar vehículos, estimar su velocidad y clasificarlos utilizando una Zona Virtual de Detección y la red neuronal YOLOv3, el factor diferencial que se puede destacar con este experimento, es que, hasta donde se pudo revisar, no se tomaron en cuenta conjunto de datos personalizados en base a diferentes escenarios diurnos y nocturnos.

El artículo "The improvement in obstacle detection in autonomous vehicles using YOLO non-maximum suppression fuzzy algorithm" descrito por (Zaghari et al., 2021) se centra en mejorar la detección de obstáculos en vehículos autónomos utilizando el algoritmo de supresión no máxima difusa (fuzzy), este trabajo está aplicado a vehículos autónomos, esto se puede considerar dentro de otro ámbito per se.

El artículo "Deep Learning for Computer Vision: A Brief Review" escrito por (Voulodimos et al., 2018) proporciona una revisión concisa sobre el aprendizaje profundo (deep learning) aplicado a la visión por computadora, este trabajo funge como base de conocimiento para establecer el know how del desarrollo de este trabajo porque aborda conceptos sobre el aprendizaje profundo que se deben tener en cuenta en nuestro caso práctico.

El artículo de (Tan et al., 2021) se basa en la comparativa de los mismos modelos seleccionados por nuestra parte pero con la diferencia de que hasta donde sabemos en este artículo no hace referencia a la segunda mayor característica de estos modelos que

es el reconocimiento, este trabajo se centra en la clasificación que se puede encontrar con estos modelos y adicionalmente las versiones con las que están construidas los experimentos son distintas, en este trabajo adicional usa SSD con VGG16 como extractor de características, a diferencia de nosotros que tenemos las capas de la columna vertebral con MobilNet, que es un nuevo modelo actualizado y que presenta las ventajas de detección en función del tiempo, por su modelo de arquitectura.

Dentro del experimento de (Almeida et al., 2022) aborda la problemática de la detección y conteo vehicular usando OPEN CV como marco de trabajo y haciendo un desarrollo iterativo tomando los algoritmos Mixture of Gaussian (MOG), k-Nearest Neighbor (KNN) y Geo-metric Multigrid.

Debido a que se han realizado modificaciones en la estructura de cada red neuronal en sus versiones posteriores, así como se han buscado combinaciones de modelos como el caso de RetinaNet con ResNet50 y Single Shot detector con Mobilnet, existe una nueva gama de posibilidades en el procesamiento de detección y conteo de vehículos en tiempo real. Por lo tanto, es deseable una interpretación nueva y mejorada del desempeño de estas técnicas.

2.2. Acción

Herramientas de Desarrollo y Configuración de Ambiente de Trabajo

La selección de herramientas del experimento se realizó por la naturaleza de este, parte fundamental del experimento era poder analizar en tiempo real el comportamiento de la detección y conteo de vehículos, esto nos orientó a seleccionar arquitecturas que sean de una sola etapa principalmente por a la relación entre rapidez y eficiencia, así como:

- Eficiencia: Los detectores de una etapa son generalmente más rápidos en términos de velocidad de detección, ya que realizan la detección directamente en una sola pasada de la red neuronal.

- Simplicidad: Suelen ser más sencillos en términos de arquitectura y proceso de detección, ya que no requieren etapas adicionales de propuestas de regiones o refinamiento.

- Detección directa: Detectan objetos y sus ubicaciones directamente en la imagen sin necesidad de etapas intermedias. Esto puede resultar en una mayor capacidad para detectar objetos pequeños y densos.

Siguiendo el orden cronológico, así como la evolución que ha tenido cada arquitectura se contempló las que contengan una mayor base bibliográfica y documental en este caso YOLO, SSD, RN.

En esta etapa de acción, se implementó el caso práctico basado en diferentes enfoques y técnicas para mejorar la precisión del sistema de detección. Se hizo un entorno tipo cuaderno de cada arquitectura, se exploraron alternativas para anexar otras arquitecturas en la etapa de extracción de características, se exploró la documentación oficial de cada una para realizar el experimento lo más preciso posible.

A continuación, se explica de manera breve la tecnología y las herramientas seleccionadas para el presente trabajo, para que los lectores puedan entender más claramente las siguientes secciones.

Es bien conocido por todos, que el trabajo con algunas de las ramas de la inteligencia artificial requiere de un alto poder de procesamiento y requerimientos de hardware; En el artículo de Vivienne (Sze et al., 2017) , aborda los desafíos y oportunidades asociados con el hardware utilizado en el aprendizaje automático. Proporciona una visión general de las demandas de hardware en esta área y destaca las soluciones especializadas que se están desarrollando para mejorar el rendimiento y la eficiencia de los modelos de aprendizaje automático. sin embargo, gracias a la revolución de la computación en la nube que brinda recursos como un modelo de entrega de servicios de TI a través de Internet. En lugar de tener que poseer y administrar infraestructuras físicas, como servidores y centros de datos, las empresas y los usuarios pueden acceder a recursos computacionales, almacenamiento y software a través de proveedores de servicios en la nube.

Debido a esta flexibilidad, se optó por aprovechar este enfoque usando Google Colaboratory que, es una plataforma en la nube desarrollada por Google que permite a los usuarios escribir y ejecutar código en lenguaje Python de forma interactiva. Colab proporciona un entorno de cuaderno colaborativo basado en Jupyter Notebooks, lo que significa que los usuarios pueden crear documentos que contienen código, texto descriptivo y visualizaciones interactivas. Una de las características destacadas de Colab es su

capacidad para ejecutar código en la nube, lo que significa que no es necesario configurar ni ejecutar código en una máquina local. Además, Colab ofrece acceso gratuito a recursos computacionales, como CPU, GPU y TPU, lo que permite realizar cálculos intensivos y entrenar modelos de aprendizaje automático sin la necesidad de una infraestructura de hardware dedicada. En resumen, Google Colaboratory es una herramienta poderosa que proporciona un entorno de desarrollo interactivo en la nube para trabajar con lenguaje Python, lo que facilita la colaboración y el acceso a recursos computacionales de alto rendimiento en concordancia con (Chhabra & Singh, 2022).

El motivo por el cual se optó por utilizar estas herramientas en la web, en el caso de Google Colaboratory es para aprovechar el hardware “on demand” que se puede adquirir a costos accesibles para entrenar los modelos sin tener necesariamente que comprar recursos además que al ser un entorno virtual no requiere instalación en la máquina hospedadora, lo hace sencillo de utilizar y transportable, lo mismo que sucede en la etapa de etiquetado de imágenes, al usar una herramienta en la web que mostró buen desempeño al realizar esta tarea, brinda flexibilidad al momento de desarrollar el caso práctico.

Para los experimentos del presente trabajo, se contrató 100 unidades de procesamiento (on demand) en esta plataforma de manera adicional, para mejorar los recursos que nos ofrece la capa gratuita.

Por otra parte, y en concordancia con (Brigato et al., 2022), entrenar un modelo de visión por computadora con tu propio dataset ofrece ventajas como la personalización, el control de calidad, la adaptabilidad, la privacidad, un mejor rendimiento y contribuciones a la investigación en el campo. Estas ventajas hacen que el uso de un dataset como menciona (Li et al., 2020) personalizado sea valioso para aplicaciones específicas y escenarios donde se requiere un alto grado de precisión y adaptabilidad.

Para este experimento se construyó un dataset de 300 imágenes tomadas de videos que no poseen derechos de autor de las diferentes cámaras de seguridad en puentes y autopistas de distintos ángulos, perspectivas y condiciones meteorológicas. Adicionalmente se aprovechó del ecosistema de la plataforma que permite la interconexión con sus servicios de almacenamiento para poder almacenar los checkpoints del entrenamiento, métricas y disposición de recursos read/write.

El etiquetado se refiere al proceso de etiquetar los datos de entrenamiento para indicar la clase o la categoría a la que pertenecen. Dentro del trabajo “Types for Information Flow Control: Labeling Granularity and Semantic Models” de (Rajani & Garg, 2018), En cuanto a los modelos semánticos, el artículo examina cómo se representan y gestionan las etiquetas de información en relación con el significado semántico de los datos. Existen diferentes tipos de etiquetado que se utilizan en las redes neuronales para este experimento son:

Etiquetado de clase única: En este tipo de Etiquetado, cada dato de entrenamiento se etiqueta con una única clase o categoría. Por ejemplo, en un problema de clasificación de imágenes de vehículos livianos y vehículos pesados, cada imagen se etiquetaría como "carro" o "moto".
Etiquetado de detección de objetos: En la detección de objetos, además de etiquetar la clase a la que pertenece un objeto en una imagen, también se etiqueta su ubicación mediante cajas delimitadoras. Esto implica etiquetar la clase del objeto y las coordenadas (x, y) de la caja delimitadora que lo rodea.
Etiquetado de segmentación semántica: En este tipo de etiquetado, se etiqueta cada píxel de una imagen para indicar a qué clase o categoría pertenece. Esto permite realizar una segmentación precisa de la imagen en diferentes regiones semánticas.

Se usó como emulación del proceso de detección y conteo en tiempo real de fragmentos de video capturados por drones de 30 seg, 60 seg, y 120 seg para poder recopilar los resultados de los experimentos.

Se tomó muy en cuenta las anotaciones descritas en el artículo "Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping" escrito por (Dodge et al., 2020) se centra en la técnica de ajuste fino de modelos de lenguaje pre-entrenados. Los autores exploran diferentes aspectos del proceso de ajuste fino y cómo afectan al rendimiento del modelo. Donde, investigan el impacto de las inicializaciones de peso en el ajuste fino. Analizan diferentes métodos de inicialización y cómo afectan a la capacidad del modelo para capturar información relevante durante el ajuste fino.

Basándonos que estos 3 modelos son justamente redes que ya han sido entrenadas y a las que reentrenamos para ajustar sus pesos y enfocarlos en la clase “vehículo”, que cabe recalcar tomamos como palabra clave vehículo que contempla automotor liviano, automotor pesado y motocicletas.

Tal y como presenta en su artículo, la estrategia para contar vehículos se toma como en este experimento, delimitando una línea meta en un sentido denominado umbral, cualquier clase que pase por este umbral con un nivel de certeza mayor al 60% se toma como vehículo contado para recabar la matriz de confusión.

La herramienta web makesense.ai, que usamos para etiquetar las regiones con la clase que vamos a entrenar, en nuestro caso llamamos a la clase “vehiculo”, esta herramienta permite exportar estas anotaciones en formato XML y CSV. Como se muestra en la figura 9.

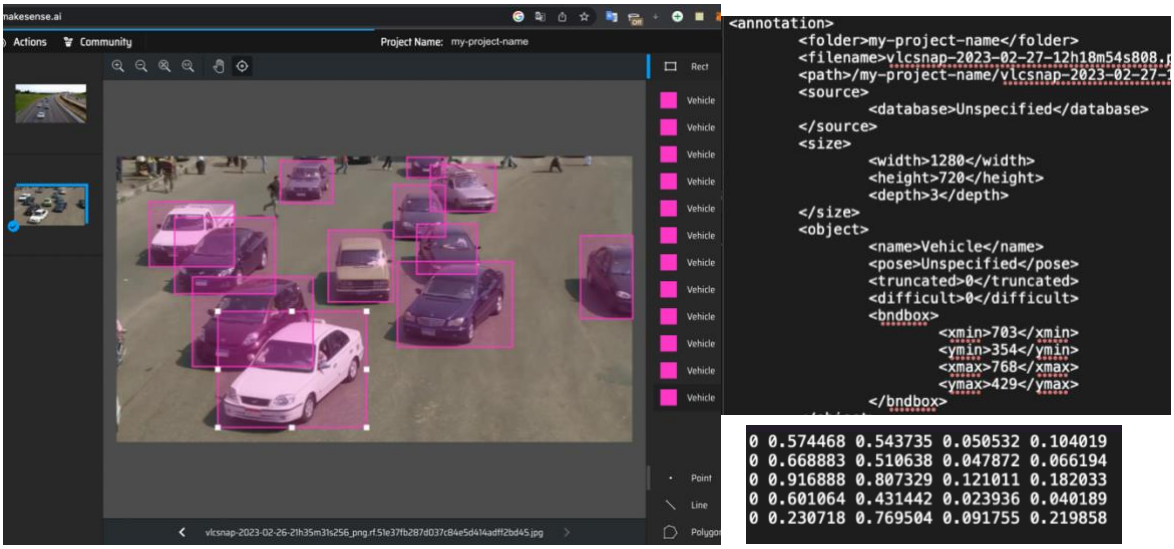

Figura 9 - Herramienta MakeSense.ai anotaciones en XML y en CSV

Una vez terminada la tarea de etiquetado de imágenes, se exporta en ambos formatos, y creamos un nuevo cuaderno con nuestra cuenta de Google en Google Colaboratory, aquí nos proporcionan un entorno de desarrollo en la nube y lo conectamos con nuestro Google Drive para que funcione como proveedor de almacenamiento como se muestra en la figura 10, el entorno se va a ver de lado izquierdo el directorio de archivos que responden a comandos de Linux y de lado derecho, el entorno de ejecución del cuaderno que tiene las siguientes versiones de Python 3.10.X.

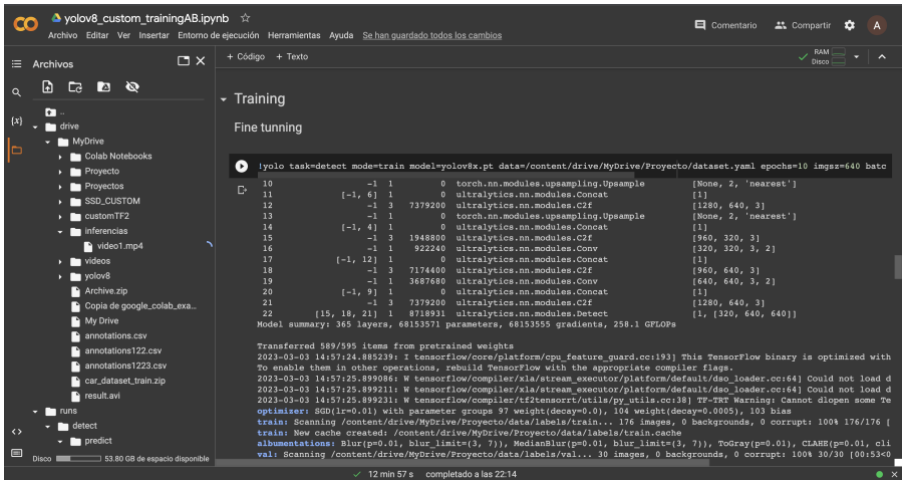

Figura 10 - Entorno de desarrollo Google Colaboratory

Para las tres arquitecturas, hemos tomado modelos pre-entrenados que vamos a reentrenar con nuestro set de datos customizado, de esta manera podemos agilitar el tiempo que requiere de entrenamiento porque ya tiene pesos iniciados, en el caso de YOLOv8, tiene su documentación oficial actualizada donde se encuentra todos los parámetros necesarios para el entrenamiento, nosotros trabajaremos con cada modelo durante cincuenta épocas. Para empezar, como proyecto conceptual, intentaremos obtener los mejores resultados posibles con la menor cantidad de capacitación disponible. Incluso 50 épocas, ya que tenemos casi 300 imágenes, tomarán bastante tiempo para entrenar y deberían dar resultados satisfactorios.

En todos los experimentos, estableceremos el tamaño del lote en 8 para tener una comparación justa entre los modelos. Durante el entrenamiento, estableceremos el tamaño de la imagen en 1280 x 768 porque los baches pueden ser bastante pequeños en algunas imágenes. Podemos esperar mejores resultados en comparación con el entrenamiento predeterminado de resolución de imagen 640, aunque esto requerirá más tiempo.

Luego de terminar con el entrenamiento, se valida y se continua con la parte del conteo.

El objetivo del seguimiento de objetos múltiples (MOT) es estimar los límites y las identidades de los objetos en los videos. Un enfoque simple para esto implica varios pasos. En primer lugar, se detectan los límites de los objetos (en este caso, personas) en todos los fotogramas utilizando un algoritmo de detección de objetos. En la imagen de ejemplo, se resaltan en amarillo los límites de las personas detectadas en cada fotograma, junto con

un umbral de confianza asociado. Este proceso se repite para cada fotograma del video. Luego, se utiliza un algoritmo de seguimiento, como ByteTrack, para asociar los límites de detección a lo largo de los fotogramas. En la imagen de ejemplo, después de aplicar el algoritmo de seguimiento, cada objeto (persona) es asignado a una identificación de seguimiento. Los objetos con la misma identificación de seguimiento se muestran en el mismo color tal como muestra la figura 11.

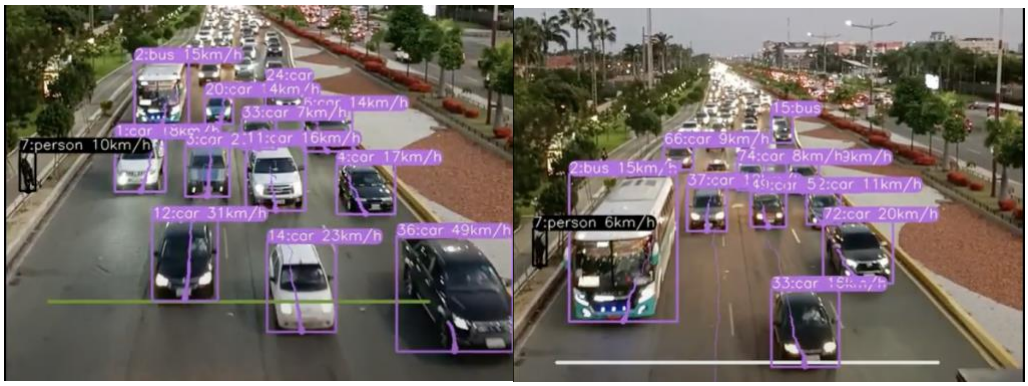

Figura 11 - Inferencia de Conteo y Seguimiento YOLOv8

En el caso de SSD, se repite en esencia el mismo enfoque, vamos a reentrenar un modelo pre-entrenado el modelo específicamente es SSD300 con nuestro conjunto de datos propios, en este caso usamos el repositorio propio de weiliu89/caffe, adicionalmente en con esta arquitectura es necesario usar TensorFlow debido a la capacidad de TensorFlow para realizar cálculos eficientes en redes neuronales, su amplia comunidad de desarrolladores y su versatilidad para trabajar en diferentes plataformas y dispositivos y las anotaciones tiene que estar en un formato específico denominado TFRecord es una forma eficiente y optimizada de almacenar datos en TensorFlow.

Como uno de nuestros objetivos es exponer a las arquitecturas a las mismas conficiones de entrenamiento, aquí también se ha tomado 50 épocas y un batch size igual a 8, (similar que en YOLO) teniendo finalmente un modelo que presenta la siguiente inferencia, Adicionalmente, para la parte del conteo se tomó la misma estrategia, se creó una línea delimitadora que cuando tiene un porcentaje superior al 60% de inferencia de que se trata de la clase vehículo, aumenta en uno el contador tal como presenta la figura 12.

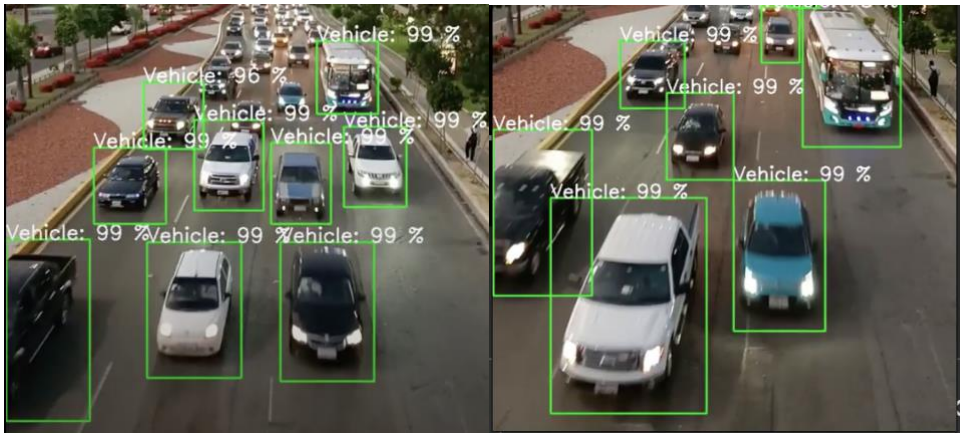

Figura 12 - Inferencia Conteo y Detección de Single Shot Detector

Dentro del caso de estudio de RetinaNet, partimos del repositorio original del primer artículo donde lo mencionan (fizyr/keras-retinanet).

En este apartado se usó los mismos hiperparámetros para el entrenamiento (50 épocas y 8 en batch size) y un modelo que ha sido entrenado previamente para hacerle un ajuste fino que permita detectar nuestras nuevas clases.

Esta arquitectura también se apoya en Tensorflow como su biblioteca de convoluciones y luego de entrenarlo deja una inferencia como la muestra la ilustración que la diferencia es que debe reducir el número de fotogramas para tener una mejor precisión.

Para la fase de conteo, utilizamos el mismo procedimiento, Se dibujó una línea de meta como referencia y se verificó si los objetos detectados intersecan o se cruzan con esta marca, Esto se consigue mediante la comparación de las coordenadas de los objetos con la posición de la señal incrustada. como se muestra en la figura 13.

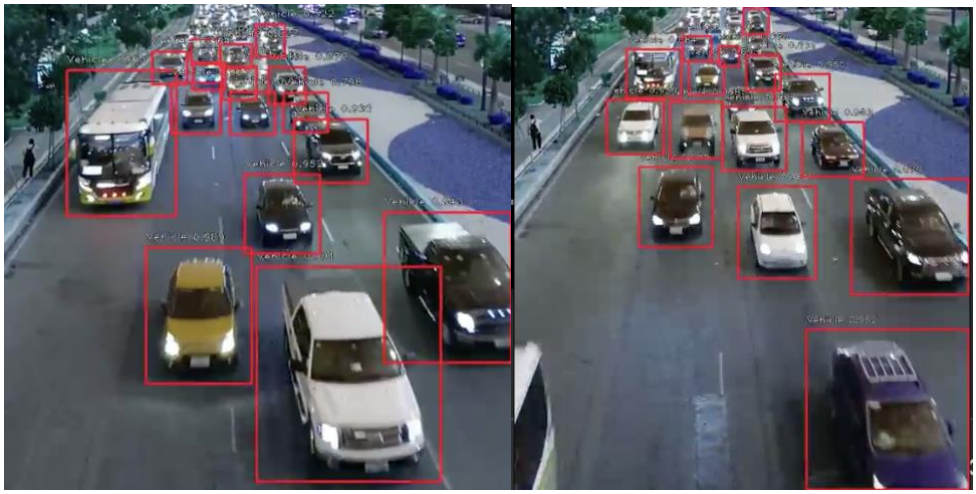

Figura 13 - Inferencia RetinaNet Conteo y Detección.

3. RESULTADOS Y DISCUSIÓN

Después de la implementación, se llevó a cabo la etapa de observación y reflexión, donde tabulamos los resultados obtenidos y analizamos los datos recopilados, reflexionando sobre los resultados.

Asimismo, para ofrecer una visión imparcial de los desempeños de los 3 modelos, adoptamos un enfoque cuantitativo basado en métricas que son comúnmente utilizadas para evaluar estos modelos, en este caso, precisión, recuperación y puntuación F1.

La precisión de las predicciones está determinada por la proporción de verdaderos positivos frente a los negativos, La recuperación es la relación entre el número de verdaderos positivos y el número total de clases y se le da peso a un promedio de precisión y recuperación para el puntaje F1 como lo comenta (Padilla et al., 2020) en su artículo.

Precisión = (Positivo verdadero)/(Positivo verdadero + Positivo falso)
Recuperación = (Positivo verdadero)/(Positivo verdadero + Negativo falso)
Puntuación F1 = (Precisión × Recuperación)/[(Precisión+ Recuperación)/2]

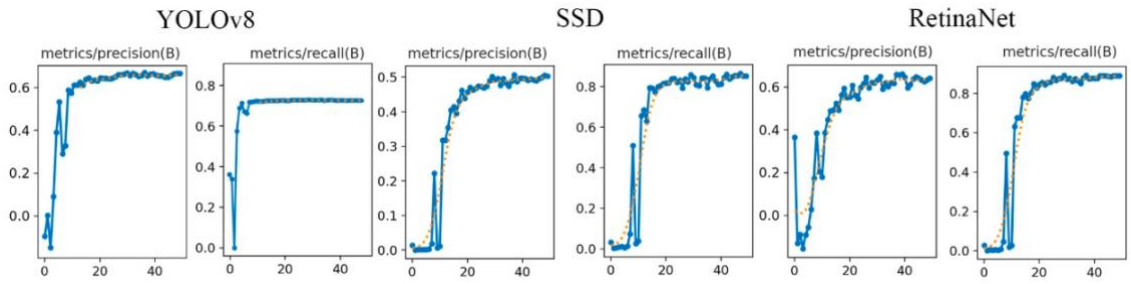

Figura 14 – Curvas de precisión y recuperación de los resultados obtenidos

En las siguientes tablas, podemos observar los resultados obtenidos para 3 videos con duraciones de 30 segundos, 60 segundos y 90 segundos, respectivamente. Las columnas muestran el error absoluto, que representa el número de vehículos que no fueron detectados, seguido del error relativo, que es el porcentaje del error absoluto en comparación con el número total de vehículos circulando en los videos. También tenemos falsos positivos, que ocurren cuando el algoritmo o modelo de detección identifica incorrectamente un objeto que no está presente en la imagen o video analizado. Por otro lado, la última columna representa falsos negativos, que ocurren cuando el algoritmo o modelo de detección no logra identificar correctamente un objeto que está presente en la imagen o video analizado.

Finalmente se han agregado gráficos de métricas comunes en este tipo de modelos que obedecen al desempeño que se obtuvo a lo largo de la etapa de experimentación.

3.1. Resultados

Tabla 3 - Cuadro de resultados de inferencias SSD, YOLO, RN en un video de 30 segundos

Video 1 30''				
Algoritmo	% Error Absoluto	% Error Relativo	Falso Positivo	Falso Negativo
YOLO v8	5	11.46%	0	2
SSD + Mobilnet	9	16.55%	1	8
RetinaNet + ResNet50	3	9%	0	2

Tabla 4 - Cuadro de resultados de inferencias SSD, YOLO, RN en un video de 60 segundos

Video 2 60''				
Algoritmo	% Error Absoluto	% Error Relativo	Falso Positivo	Falso Negativo
YOLO v8	5	8.86%	0	6
SSD + Mobilnet	13	19.23%	3	14
RetinaNet + ResNet50	6	12.01%	1	4

Tabla 5 - Cuadro de resultados de inferencias SSD, YOLO, RN en un video de 90 segundos

Video 3 90''				
Algoritmo	% Error Absoluto	% Error Relativo	Falso Positivo	Falso Negativo
YOLO v8	8	9.11%	1	6
SSD + Mobilnet	12	17.29%	2	19
RetinaNet + ResNet50	12	17.29%	2	6

Tabla 6 - Cuadro de resultados, precisión, recuperación, f1, mAP

Algoritmo	Precisión%	Recuperación%	F1%	MAP%
YOLO v8	66.31%	75.19%	70.47%	79.01%
SSD + Mobilnet	59.98%	84.83%	70.27%	83.32%
RetinaNet + ResNet50	61.69%	88.03%	72.54%	87.54%

3.2. Discusiones

Existen 2 problemáticas alrededor de este experimento, la primera es la detección vehículos en tiempo real; la segunda es el conteo de los vehículos que se encuentran en cuadro. Dividimos el experimento en tres etapas, Recolección de datos y Reentrenamiento de modelos; Detección de Vehículos; Conteo de Vehículos. Inicialmente tomamos cámaras de seguridad de puentes y avenidas, se extrajo imágenes de los videos en diferentes posiciones, de diferentes ángulos y en diferentes condiciones de luz para obtener un set de datos variado que posteriormente etiquetaríamos con la herramienta online makesense.ai; reentrenamos a los modelos de YOLOv8 de ultralytics, SSD + Mobilnet

como capa de extracción de características y RetinaNet + ResNet50 con este set personalizado de datos y apalancándose de la herramienta de Google Colaboratory.
Dentro de la implementación; YOLOv8 al ser el modelo que tiene actualizaciones periódicas es muestra una ventaja significativa con respecto a los otros dos modelos.

En nuestro estudio, hemos evaluado y analizado los modelos de detección de objetos YOLO y MobileNet SSD en diversos escenarios. Cada uno de estos modelos tiene características únicas y ha demostrado éxito en diferentes aplicaciones.
En cuanto a la precisión, hemos encontrado que YOLO ofrece resultados más precisos en comparación con MobileNet SSD. Sin embargo, en términos de velocidad de detección, MobileNet SSD supera a YOLO, siendo capaz de realizar detecciones más rápidas.
MobileNet SSD, también conocido como SSD, es un detector único que puede manejar múltiples clases y ofrece una velocidad de detección más rápida en comparación con los detectores progresivos anteriores, como YOLO. Además, se ha observado que MobileNet SSD tiene una precisión comparable a técnicas más lentas que utilizan diseños y agrupaciones de regiones rápidas, como los R-CNN más rápidos.

Es importante tener en cuenta que MobileNet SSD funciona mejor en casos donde el tamaño del objeto es pequeño como se muestra en la figura 15, aunque puede haber una ligera disminución en el rendimiento en comparación con YOLO. En resumen, la elección entre YOLO y MobileNet SSD depende de los requisitos específicos de cada aplicación. YOLO se recomienda para objetos pequeños, mientras que MobileNet SSD es más adecuado cuando se prioriza la velocidad de detección y el tamaño del objeto es pequeño.

Figura 15 - Desempeño de detección de objetos pequeños

RetinaNet muestra una precisión de alrededor de 86%, sin embargo, el procesamiento de fotogramas por segundo limitado, de alrededor de 22fps hace que sea un método poco

elegible para este estudio, tomando en cuenta que se busca mostrar el mejor rendimiento en tiempo real. Para el segundo enunciado, se buscó empalmar cada solución con su respectiva solución de tracking y hacer un algoritmo de conteo; Yolov8 con ByteTrack y Supervision, y se tuvo un mejor desempeño significativo con respecto a SSD con tensorflow_object_counting_api.

Como futuros trabajos, se puede explorar un conjunto de iteraciones de modelos complementarias y compararlos, por ejemplo, RetinaNet asociarlo con otros modelos como MobileNet o EfficientNet para extracción de características para mejorar el desempeño.

4. CONCLUSIONES

Hemos sometido a las arquitecturas de detección de objetos en una sola etapa YOLOv8, SSD y RN a un ajuste fino con nuestro propio conjunto de datos de vehículos para analizar el rendimiento que tiene en la detección y conteo de vehicular, usamos makesense.ai para etiquetar las imágenes y Google Colab para el reentrenamiento de las arquitecturas. Se complementó SSD300 con mobilnet y se obtuvo un mAP de alrededor del 83% pero a 45 fotogramas por segundo y presentó un mayor error relativo, RetinaNet se usó con ResNet-50 obteniendo alrededor de un mAP de 87% a 22 fotogramas por segundo, debido a que este problema es directamente proporcional a la velocidad de los fotogramas para poder detectar en tiempo real, YOLOv8 puede sacrificar un poco de mAP de alrededor de 79% por la velocidad de detección de 60 fotogramas por segundo, adicionalmente la comunidad de soporte es grande y de una implementación significativamente más fácil, siendo la mejor alternativa para este propósito.

Finalmente, consideramos que la pregunta se transforma de simplemente seleccionar una arquitectura categóricamente a encontrar la mejor combinación de algoritmos y modelos que potencien el rendimiento de esta tarea y, además, en este trabajo no se profundizó en el hardware donde se aloja la solución, cosa que se puede tener en cuenta en siguientes trabajos.

REFERENCIAS BIBLIOGRÁFICAS

Almeida, J., Guamán, S. & Yoo, S. G. (2022). Vechicle Counting System in Urban Areas: A Practical Case. *2022 IEEE 7th International Conference for Convergence in Technology (I2CT)*, 1–6. https://doi.org/10.1109/I2CT54291.2022.9823982

Altrichter, H., Kemmis, S., McTaggart, R. & Zuber-Skerritt, O. (2002). The concept of action research. *The Learning Organization*, *9*(3), 125–131. https://doi.org/10.1108/09696470210428840

Bahri, F., Shakeri, M. & Ray, N. (2020). Online Illumination Invariant Moving Object Detection by Generative Neural Network. *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*. https://doi.org/10.1145/3293353.3293369

Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A. & Ouni, K. (2018). *Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3*.

Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K. & Ghayvat, H. (2021). *electronics CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope*. https://doi.org/10.3390/electronics10202470

Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*.

Brigato, L., Barz, B., Iocchi, L. & Denzler, J. (2022). *Image Classification with Small Datasets: Overview and Benchmark*. https://doi.org/10.1109/ACCESS.2022.3172939

Chhabra, S. & Singh, A. K. (2022). *A COMPREHENSIVE VISION ON CLOUD COMPUTING ENVIRONMENT: EMERGING CHALLENGES AND FUTURE RESEARCH DIRECTIONS A PREPRINT*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H. & Smith, N. (2020). *Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping*. https://github.com/huggingface/

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202. https://doi.org/10.1007/BF00344251

Gómez-Huélamo, C., Conde, M. V & Ortiz, M. (2022). *Exploring Map-based Features for Efficient Attention-based Vehicle Motion Prediction*.

He, K., Zhang, X., Ren, S. & Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *CoRR*, *abs/1406.4729*. http://arxiv.org/abs/1406.4729

He, K., Zhang, X., Ren, S. & Sun, J. (2015). *Deep Residual Learning for Image Recognition*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, *abs/1704.04861*. http://arxiv.org/abs/1704.04861

Jocher, G. (2020). *YOLOv5 by Ultralytics*. https://doi.org/10.5281/zenodo.3908559

Jocher, G., Chaurasia, A. & Qiu, J. (2023). *YOLO by Ultralytics*. https://github.com/ultralytics/ultralytics

Li, X., Wei, T., Pun Chen, Y., Tai, Y.-W. & Tang, C.-K. (2020). *FSS-1000: A 1000-Class Dataset for Few-Shot Segmentation*. https://github.com/HKUSTCV/FSS-1000

Lin, C.-J., Jeng, S.-Y. & Lioa, H.-W. (2021). *A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO*. https://doi.org/10.1155/2021/1577614

Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B. & Belongie, S. J. (2016). Feature Pyramid Networks for Object Detection. *CoRR*, *abs/1612.03144*. http://arxiv.org/abs/1612.03144

Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. (2018). *Focal Loss for Dense Object Detection*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C.-Y. & Berg, A. C. (2015). SSD: Single Shot MultiBox Detector. *CoRR*, *abs/1512.02325*. http://arxiv.org/abs/1512.02325

Mahdi, F., Motoki, K. & Kobashi, S. (2020). Optimization technique combined with deep learning method for teeth recognition in dental panoramic radiographs. *Scientific Reports*, *10*. https://doi.org/10.1038/s41598-020-75887-9

Padilla, R., Netto, S. L. & da Silva, E. A. B. (2020). A Survey on Performance Metrics for Object-Detection Algorithms. *2020 International Conference on Systems, Signals and*

Image Processing (IWSSIP), 237–242. https://doi.org/10.1109/IWSSIP48289.2020.9145130

Papert, S. A. (1966). *The Summer Vision Project*. http://hdl.handle.net/1721.1/6125

Rajani, V. & Garg, D. (2018). *Types for Information Flow Control: Labeling Granularity and Semantic Models*.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*.

Redmon, J. & Farhadi, A. (2016). *YOLO9000: Better, Faster, Stronger*. http://pjreddie.com/yolo9000/

Redmon, J. & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*.

Roberts, L. G. (1963). Machine Perception of Three-Dimensional Solids. *Outstanding Dissertations in the Computer Sciences*. http://hdl.handle.net/1721.1/11589

Sze, V., Chen, Y.-H., Emer, J., Suleiman, A. & Zhang, Z. (2017). *Hardware for Machine Learning: Challenges and Opportunities*.

Tan, L., Huangfu, T., Wu, L. & Chen, W. (2021). Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification. *BMC Medical Informatics and Decision Making*, *21*(1), 324. https://doi.org/10.1186/s12911-021-01691-8

Terven, J. R. & Cordova-Esparaza, D. M. (2023). *A COMPREHENSIVE REVIEW OF YOLO: FROM YOLOV1 AND BEYOND UNDER REVIEW IN ACM COMPUTING SURVEYS*.

Voulodimos, A., Doulamis, N., Doulamis, A. & Protopapadakis, E. (2018). *Deep Learning for Computer Vision: A Brief Review*. https://doi.org/10.1155/2018/7068349

Wightman, R., Touvron, H. & Jégou, H. (2021). *ResNet strikes back: An improved training procedure in timm*.

Williams, R. I., Clark, L. A., Clark, W. R. & Raffo, D. M. (2021). Re-examining systematic literature review in management research: Additional benefits and execution protocols. *European Management Journal*, *39*(4), 521–533. https://doi.org/10.1016/J.EMJ.2020.09.007

Zaghari, N., Fathy, M., Jameii, S. M. & Shahverdy, M. (2021). The improvement in obstacle detection in autonomous vehicles using YOLO non-maximum suppression fuzzy algorithm. *The Journal of Supercomputing*, *77*(11), 13421–13446. https://doi.org/10.1007/s11227-021-03813-5

Zhang, H. & Cloutier, R. (2022). Review on One-Stage Object Detection Based on Deep Learning. *EAI Endorsed Transactions on E-Learning*, *7*, 174181. https://doi.org/10.4108/eai.9-6-2022.174181

Zhang, L., Wang, H., Wang, X., Chen, S., Wang, H., Zheng, K. & wang, H. (2021). Vehicle Object Detection Based on Improved RetinaNet. *Journal of Physics: Conference Series*, *1757*(1), 12070. https://doi.org/10.1088/1742-6596/1757/1/012070

Zhao, Z.-Q., Zheng, P., Xu, S.-T. & Wu, X. (2019). Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865

Zou, Z., Shi, Z., Guo, Y. & Ye, J. (2019). Object Detection in 20 Years: A Survey. *CoRR*, *abs/1905.05055*. http://arxiv.org/abs/1905.05055