UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CARRERA DE TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN TELECOMUNICACIONES

"Sistema de detección de señales de tránsito mediante visión por computador."

AUTOR:

CACAO GRANOBLE MIGUEL ABEL

DOCENTE TUTOR:

ING. MANUEL MONTAÑO, M.Sc

LA LIBERTAD-ECUADOR

2024

DECLARACIÓN DE DOCENTE TUTOR

En mi calidad de Docente Tutor del Trabajo de Integración Curricular**, "Sistema de detección de señales de tránsito mediante visión por computador"**, elaborado por el señor Miguel Abel Cacao Granoble, estudiante de la Carrera de Telecomunicaciones, Facultad de Sistemas y Telecomunicaciones de la Universidad Estatal Península de Santa Elena, previo a la obtención del título de Ingeniería en Telecomunicaciones, me permito declarar que, tras supervisar el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos. En consecuencia, lo considero apto en todos sus aspectos y listo para la ser evaluado por el docente especialista.

Atentamente

Ing. Manuel Montaño Blacio, M.Sc.

DOCENTE TUTOR

DECLARACIÓN AUTORÍA DEL ESTUDIANTE

El presenta trabajo de Integración Curricular con el título **"Sistema de detección de señales de tránsito mediante visión por computador"**, declaro que la concepción, análisis y resultados son originales a la activada educativa en el área de Telecomunicaciones.

Atentamente

Sr. Cacao Granoble Miguel Abel

C.I. 1752503407

DECLARACIÓN DE DOCENTE ESPECIALISTA

En mi calidad de Docente Especialista del Trabajo de Integración Curricular, **"Sistema de detección de señales de tránsito mediante visión por computador"**, elaborado por el señor Miguel Abel Cacao Granoble, estudiante de la Carrera de Telecomunicaciones, Facultad de Sistemas y Telecomunicaciones de la Universidad Península de Santa Elena, previo a la obtención del título de Ingeniería en Telecomunicaciones, me permito declarar que, tras supervisar el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos. En consecuencia, lo considero apto en todos sus aspectos y listo para la sustentación del trabajo.

Atentamente

Ing. Vladimir Garcia Santos, Mgtr.

DOCENTE ESPECIALISTA

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Rovira Jurado, Ph.D.
DIRECTOR DE LA CARRERA

Firmado electrónicamente por:
VLADIMIR ISRAEL
GARCIA SANTOS

Ing. Vladimir Garcia Santos, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
MANUEL ASDRUVAL
MONTANO BLACIO

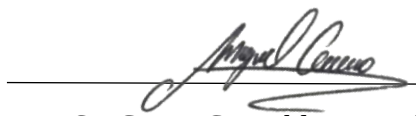
Ing. Manuel Montaño Blacio, M. Sc.
DOCENTE TUTOR GUÍA

Ing. Corina Gonzabay De la A, Mgtr.
SECRETARÍA

DECLARATORIA DE RESPONSABILIDAD

Quien suscribe, MIGUEL ABEL CACAO GRANOBLE con C.I. 1752503407, estudiante de la carrera de Telecomunicaciones, declaro que el trabajo de Titulación presentado a la unidad de Integración Curricular cuyo tema es **"Sistema de detección de señales de tránsito mediante visión por computador"**, corresponde y es de exclusiva responsabilidad del autor y pertenece al patrimonio intelectual de la Universidad Estatal Península de Santa Elena.

Atentamente



Sr. Cacao Granoble Miguel Abel

C.I. 1752503407

AGRADECIMIENTO

En primer lugar, quiero expresar mi más sincero agradecimiento a Dios, fuente de sabiduría y guía en cada paso de mi vida. Su amor incondicional y misericordia han sido mi mayor inspiración y fortaleza durante toda esta travesía académica. A mi madre, Elsa, le debo un agradecimiento especial. Desde el inicio de mi carrera, ella ha sido mi mayor apoyo, inculcándome buenos valores y la importancia de la perseverancia. Su amor y dedicación han sido el faro que iluminó mi camino, y estoy eternamente agradecido por su influencia positiva en mi vida.

En segundo lugar, quiero agradecer a mi tutor, el Ing. Manuel Montaño Blacio, por su orientación invaluable en la elaboración de este proyecto. Su experiencia y dedicación fueron fundamentales para guiarme paso a paso en este proceso, brindándome conocimientos cruciales y perspectivas enriquecedoras. También debo reconocer y agradecer al Ing. Vladimir García Santos, por su constante apoyo a lo largo de la carrera. Sus conocimientos compartidos generosamente han sido una fuente inestimable para mi desarrollo académico y profesional.

Por último, pero no menos importante, quiero expresar mi gratitud a mis compañeros de clase y a todo el equipo de la facultad de telecomunicaciones. Su colaboración, compañerismo han enriquecido mi experiencia universitaria. A todos quienes forman parte de esta comunidad, gracias por contribuir al éxito de este camino académico.

Miguel Abel Cacao Granoble

DEDICATORIA

A Dios, mi guía y fortaleza en cada paso de mi vida, le dedico este logro. Gracias por darme fuerzas en los momentos difíciles y por ser mi refugio constante. Este trabajo es un testimonio de tu amor incondicional y sin ti nada de esto sería posible.

A mi querida madre, Elsa, le dedico este esfuerzo con profundo agradecimiento. Tu apoyo incondicional y amor han sido el cimiento sólido sobre el cual construí mi camino. Tus sacrificios y enseñanzas han dejado huellas indelebles en mi corazón y han sido la luz que ha guiado cada paso. Este logro lleva consigo tu sabiduría y la gratitud de un hijo que reconoce el valor inmenso de tu presencia en su vida.

A mis queridos hermanos, quienes han sido faros de orientación en mi carrera, les agradezco sinceramente. Sus consejos sabios y apoyo constante han sido fundamentales en mi crecimiento personal y profesional. A través de risas compartidas, desafíos superados y momentos de reflexión, hemos forjado un vínculo inquebrantable que enriquece mi vida.

A todo el personal de la facultad, profesores, compañeros y administrativos, les dedico este trabajo como un reconocimiento a la comunidad que ha sido parte fundamental de mi formación académica. Gracias por compartir conocimientos, fomentar el aprendizaje y crear un ambiente propicio para el crecimiento.

Miguel Abel Cacao Granoble

RESUMEN

El proyecto tiene como objetivo principal desarrollar un modelo de aprendizaje automático especializado en la detección y reconocimiento de señales de tránsito verticales focalizado en el contexto de Ecuador. Este proceso se estructura mediante una metodología compuesta por cuatro fases. En la primera fase, se realiza una investigación detallada de bases teóricas para la comprensión de las técnicas y métodos de aprendizaje automático. La fase dos se basa en las investigaciones realizadas anteriormente, aquí se identifica características claves de varios métodos de detección de objetos, entre los cuales destacan Yolo y Teachable Machine, debido a sus capacidades y requerimientos computacionales. En la fase tres se establecen herramientas, parámetros y configuraciones de entrenamiento. Además, se crea una base de datos específica que incluye un conjunto de imágenes inéditas capturadas en puntos estratégicos de la provincia de Santa Elena. Por último, en la fase cuatro se evalúa el rendimiento del modelo utilizando las herramientas y métricas de validación.

El proyecto no solo se centra en la construcción de un modelo óptimo para detectar señales de tránsito, sino que también se extiende hacia la implementación práctica utilizando sistemas embebidos con el objetivo de analizar la inferencia del modelo en dispositivos con limitaciones de procesamiento. Para ello, se ha optado por utilizar un Raspberry Pi 4 modelo B, a la cual se dota con librerías específicas para tareas de aprendizaje automático. La elección de estas librerías es esencial para aprovechar al máximo la capacidad de procesamiento y garantizar la eficiencia del modelo en el dispositivo.

Palabras Clave: detección, entrenamiento, Teachable Machine, Raspberry Pi 4, Yolo.

ABSTRACT

The main objective of the project is to develop a machine learning model specialized in the detection and recognition of vertical traffic signs focused on the context of Ecuador. This process is structured through a methodology composed of four phases. In the first phase, a detailed investigation of theoretical bases for understanding machine learning techniques and methods is carried out. Phase two is based on the research carried out previously, here key characteristics of several object detection methods are identified, among which Yolo and Teachable Machine stand out, due to their capabilities and computational requirements. In phase three, tools, parameters and training configurations are established. In addition, a specific database is created that includes a set of unpublished images captured at strategic points in the province of Santa Elena. Finally, in phase four, the performance of the model is evaluated using validation tools and metrics.

The project not only focuses on the construction of an optimal model to detect traffic signs, but also extends towards practical implementation using embedded systems with the objective of analyzing model inference on devices with processing limitations. To do this, we have chosen to use a Raspberry Pi 4 model B, which is equipped with specific libraries for machine learning tasks. The choice of these libraries is essential to make the most of the processing capacity and guarantee the efficiency of the model on the device.

Keywords: detection, training, Teachable Machine, Raspberry Pi 4, Yolo.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

ÍNDICE DE ECUACION

ÍNDICE DE ABREVIATURAS

mAP: Precisión media

P: Precisión

R: Sensibilidad

BOX_LOSS: Pérdida de cuadros

OBJ_LOSS: Pérdida de objeto

CLS_LOSS: Pérdida de clasificación

ACC: Precisión

SSD: Detector de cajas múltiples de único disparo

IoU: Intersección sobre unión

NMS: Supresión de no máximos

CCN: Redes convolucionales

SDST: Sistema de detección de señales de tránsito

VP: Verdaderos positivos

VN: Verdaderos negativos

FP: Falsos positivos

FN: Falsos negativos

INTRODUCCIÓN

En la actualidad, los modelos de aprendizaje automático han surgido como protagonistas destacados en el área de la inteligencia artificial, transformando por completo la manera que abordamos problemas complejos en diversas áreas de trabajo. Tomando como base la arquitectura del cerebro, estos modelos han demostrado su capacidad para aprender patrones y tomar decisiones sin la intervención humana de forma directa, desencadenando avances significativos en diversas disciplinas tecnológicas. Dentro del panorama del aprendizaje automático, una de las disciplinas que han transformado la capacidad de las máquinas para entender y procesar datos de manera autónoma, encontramos un campo de aplicación excepcional: la detección de señales de tráfico. Esta área juega un papel importante en la seguridad vial, donde la capacidad de interpretar y responder adecuadamente a las señales es crucial (REY & JUAN, 2021).

El siguiente tema de tesis se sumerge en el mundo del aprendizaje automático, con un enfoque específico en la detección de señales de tránsito en Ecuador. En este contexto, nos enfocamos en los cimientos teóricos e investigaciones existentes, con la mira puesta en un objetivo específico: el entrenamiento de un modelo de reconocimiento de señales de tránsito. Esto implica enseñar a las máquinas a reconocer y comprender señales de tránsito.

El desarrollo de la presente propuesta tecnológica se la realizó por medio de 4 capítulos, los mismos que se detallan a continuación:

Dentro del primer capítulo se plantea la problemática del proyecto, seguida de la descripción en donde se dan a conocer características claves de la propuesta tecnológica, se exponen los objetivos que encaminan al entrenamiento del modelo más óptimo de detección, la justificación donde se procede a establecer razones por las que es necesario llevar a cabo la

propuesta, el alcance de la propuesta tecnológica, con el objetivo de tener en conocimiento los resultados que se espera tener del trabajo mencionado.

En el segundo capítulo, se dan a conocer antecedentes investigativos que tengan relación con la temática que se va a desarrollar; de la misma manera con los fundamentos teóricos como Machine learning, redes neuronales y otros argumentos que son explicados minuciosamente en el tema proporcionado.

El tercer capítulo presenta el desarrollo de la propuesta en la que intervienen factores como materiales y entrenamiento del modelo. Dentro del contexto de materiales, se presenta la herramienta de etiquetado, dispositivo embebido para realizar la inferencia y base de datos para el entrenamiento. Además, se explican los parámetros de entrenamiento necesarios para obtener un modelo óptimo y las métricas de evaluación con las cuales se analizará la eficiencia del modelo.

Por último, en el capítulo cuatro se llevan a cabo los análisis respectivos de la eficiencia del modelo, utilizando las métricas de pérdida y curvas de predicción y confianza obtenidas del entrenamiento. Posteriormente, se realiza la inferencia del modelo en tiempo real mediante la Raspberry Pi 4 y computadora de gama media. Para finalizar, se exponen las conclusiones y recomendaciones.

CAPÍTULO I

1. Generalidades de la propuesta

En este capítulo, se abordan temas importantes para el estudio introductorio de este trabajo tales como los antecedentes de la investigación, el planteamiento del problema de estudio, además se describen cada uno de los objetivos que se desean alcanzar para el desarrollo del proyecto, la justificación y por último el alcance de la propuesta.

1.1. Antecedentes

Durante el estudio de los proyectos que han trabajado en vehículos autónomos, se ha encontrado algunos con los mismos propósitos como Sánchez et al., 2021, este proyecto se encuentra en el ámbito de Deep Learning aplicado a la conducción autónoma. Se diseñan algoritmos basados en el aprendizaje profundo, el lenguaje de programación utilizado se basa en Python con una base de datos GTSRB (Sánchez Vidal, 2021). Así mismo, el trabajo de fin de grado de Irimia, 2021, en donde se utiliza un robot educativo llamado Robobo. En él, se realiza pruebas y estudios en el área de conducción autónoma. Aplican técnicas basadas en procesamiento de imágenes y en redes neuronales artificiales, con el objetivo de ser almacenadas y después comparadas con una base de datos previamente obtenida. Se utilizó herramientas de programación basadas en las librerías OpenCV y Keras (Irimia, 2021).

En un artículo se presenta el diseño e implementación de un prototipo de un sistema implementado en un carro, en el cual se centran en la detección de las señales de tránsito (SDST). El enfoque que se le dio fue probado en un subconjunto de señales de tránsito ecuatorianas como el Pare, Velocidad y Ceda el Paso, las pruebas se realizaron en un entorno de simulación bajo diferentes condiciones de iluminación para obtener varias respuestas, dichos resultado dieron un 97% de detección en precisión global (Flores-Calero et al., 2018). Por otro lado, el proyecto de tesis de Alarcón Vargas, A. J. (2022), presenta el entrenamiento de una red neuronal capaz de

detectar una señal de tránsito y de identificarla utilizando bloques residuales. El método que se utilizó para el desarrollo y entrenamiento de la red neuronal consta de cuatro fases: definición, entrenamiento, utilización y mantenimiento de la red neuronal (Javier & Vargas, 2022).

En el proyecto llamado Obstacle detection using a 2D LIDAR system for an autonomous Vehicle, se utiliza un sistema LIDAR 2D que utiliza el LIDAR-Lite v1 para el uso de detección y reconocimiento de obstáculos. En este proyecto los datos adquiridos fueron filtrados y procesados por fusión y segmentación de puntos de datos, los resultados obtenidos muestran que los obstáculos que tiene un ancho aproximado a un 1 metro se pueden identificar a 10 metros de distancia (Catapang et al., 2016). A su vez, Arce Cigüeñas, D. M. (2017), plantea la idea de un sistema autónomo de control de tránsito adaptado a las intersecciones de avenidas más transitadas. Cuenta con un sistema de transmisión inalámbrica a sistemas aledaños y hacia una interfaz de monitoreo en donde se analiza su correcto funcionamiento. Además, para lograr identificar los tiempos de operación en cada intersección utilizan niveles de autonomía, esto facilita la recopilación de información en tiempo real (MARTIN ARCE CIGÜEÑAS, 2017).

En el proyecto de Arce Millán, K. S., & vasco Alzate, I. S. (2018). Se creó un sistema distribuido para el reconocimiento de patrones en imágenes de video, orientado a patrones que afectan de manera negativa el tráfico vehicular en la ciudad de Santiago de Cali. A partir de dichos patrones se construyó clúster basado en tecnologías de sistemas distribuidos, además se implementó un algoritmo para el reconocimiento de patrones de interferencias en el tráfico (Millán & Alzate, 2018).

En el proyecto Aranda Barjola, L. A. (2012). Se explica el diseño y desarrollo de una aplicación que realiza el proceso de secuencia de imágenes con he objetivo de captar vehículos que circulen en el mismo sentido que el turismo. Además, se analiza vehículos acordes a una

base de datos que se va creando conforme va analizando e identificando los vehículos (Aranda Barjola, 2012).

En el trabajo de Cordón Ureña, A. (2021). Propone el desarrollo y mejora de un vehículo a bajo costo utilizando el aprendizaje profundo aplicando redes neuronales convolucionales con imágenes aplicadas a un vehículo autónomo. El objetivo principal de este proyecto es el desarrollo de un vehículo capaz de detectar señales de tránsito mediante algoritmos de aprendizaje automático utilizando sistemas embebidos que simule la conducción autónoma, para la obtención de las imágenes se utiliza una cámara web a las cuales se aplican los diferentes procesos de las redes neuronales para la detección de características específicas, además, delimitar los bordes de la imagen en donde se desee detectar las señales de tránsito (A Cordón Ureña, 2021).

En el trabajo de fin de grado de REY, L. G., & JUAN, V. G. (2021). Se propone el reconocimiento de señales de tráfico mediante visión artificial y redes neuronales en concreto redes neuronales convolucionales ya entrenadas. Utilizan una base de datos como Common Objects in Context, PASCAL Visual Object Class entre otras, a fin de evitar la recopilación de imágenes y los debidos procesos de cada una de ellas. El proyecto se implementó bajo marcos de trabajo previamente entrenados como OpenCV, MultiPath, Deepmask, etc (REY & JUAN, 2021).

1.2. Planteamiento del problema

A nivel mundial las señales de tránsito cumplen la función de mantener ordenado y seguro el movimiento automovilístico en las vías públicas, cada señal de tránsito presenta una instrucción específica como el guiar, prohibir, límites de velocidad, puntos de acceso, etc. Sin embargo, el error y negligencia humana siempre está presente, el conductor en cualquier momento puede presentar un desgaste físico o simplemente distraerse por unos segundos y no

percatarse de una señal de tránsito. Según un informe de la Organización Mundial de la Salud (OMS), Ecuador es el segundo país de latino América con unas elevadas cifras de accidentes de tránsito. La Agencia Nacional de Tránsito (ANT), en conjunto con el Instituto Nacional de Estadísticas y Censos, registra que aproximadamente cada 20 minutos se presenta un nuevo accidente de tránsito (Chicaiza, 2017).

Actualmente, la tecnología automovilística presente en Ecuador no cuenta con sistemas automatizados que alerten sobre la presencia de una señal de tránsito o que limiten la velocidad del auto. Como medida de solución, se ha optado por fotorradares, estos dispositivos electrónicos cumplen con la función de alertar al conductor a la velocidad que circula dentro de una vía (Palacios, 2015), dando resultados poco satisfactorios. Durante los últimos años el área automovilística se ha innovado constantemente con el objetivo de crear autos totalmente autónomos eliminando el factor humano el cual es causante de los accidentes de tránsito.

Empresas como Tesla Motors, Mercedes Benz, Audi y BMW, también otras que están más enfocadas en las tecnologías de información como: Google, Uber y Baidu han logrado automatizar y diseñar algoritmos capaces de tomar decisiones y reconocer señales de tránsito. La tecnología de reconocimiento y detección de señales de tránsito no se encuentra disponible para autos de gama baja del Ecuador como Chevrolet, Mazda, Toyota, etc (Buitrago et al., n.d.).

Esto se ha convertido en una desventaja y un reto para el área de ingenierías en relación a los países desarrollados y sus autos de alta gama con sistemas automatizados. Aunque el país ha optado por opciones rentables como los fotorradares, la necesidad de un sistema donde ayude a la percepción visual del conductor durante la circulación en las vías es cada vez mayor debido a la demanda tecnológica que existe en la actualidad a nivel mundial.

1.3. Descripción del proyecto

La propuesta tecnológica tiene como enfoque principal la detección y reconocimiento de las señales verticales de tránsito en un entorno de pruebas de conducción utilizando visión por computadora. Para ello, se utilizará algoritmos de aprendizaje automático orientado a la detección de objetos con el objetivo de que el sistema tenga la capacidad de detectar la señal de tránsito dentro de un video en tiempo real.

La etapa inicial implica la creación de un conjunto de imágenes en donde estén las señales que se desean detectar, en esta etapa se escogen las señales de tránsito más importantes del Ecuador, de este modo se descartan señales que no son muy recurrentes en la vía pública. Además, se utilizará herramientas de etiquetado para asignar etiquetas o categorías a datos en forma de texto. Este proceso es esencial para generar modelos de aprendizaje automático ya que proporciona ejemplos anotados que les permite aprender patrones y realizar predicciones.

En la etapa final, se llevará a cabo el entrenamiento del modelo en donde se tomará en cuenta ciertos aspectos como: precisión y confianza. Estos aspectos definirán el modelo más óptimo para la detección de señales de tránsito, para ello, se realizará un estudio de algunos modelos de aprendizaje automático más utilizados orientados a la detección de objetos. El proyecto es una herramienta de prácticas experimentales, debido a eso el dispositivo se desarrolla utilizando sistemas embebidos y módulos de fácil acceso de acuerdo a las necesidades del investigador y de los resultados que necesite obtener.

1.4. Objetivos

1.4.1. Objetivo General

- Diseñar un sistema de reconocimiento de señales de tránsito mediante visón por computadora para la percepción visual del conductor.

1.4.2. Objetivos Específicos

- Analizar técnicas y algoritmos de reconocimiento de imágenes mediante el estado del arte para entrenar el modelo más óptimo de acuerdo a los requisitos de la investigación.
- Entrenar un modelo de reconocimiento para detectar 20 señales verticales de tránsito.
- Evaluar el rendimiento del sistema tomando en consideración el error de la detección.

1.5. Justificación

La tecnología automovilística en Ecuador, carece de sistemas automatizados que puedan ayudar a la percepción visual del conductor. Esta brecha tecnológica es una desventaja en comparación a países desarrollados y sus carros autónomos. Sin embargo, esta brecha puede tratarse en cierta parte, mediante la implementación de modelos y algoritmos de detección de objetos. La introducción de estos algoritmos como Yolo, Teachable Machine, RetinaNet, entre otros. Proporcionará un impulso tecnológico en el área automovilística abriendo camino a nuevas implementaciones y sistemas de detección en las vías públicas.

Por lo tanto, el eje principal de este proyecto es realizar el entrenamiento de un modelo de aprendizaje automático orientado a la detección de objetos y evaluar su capacidad tomando en cuenta su error en la detección. Para lograr esto, se llevará a cabo un estudio detallado de las diferentes técnicas y modelos de reconocimientos de imágenes, así como la precisión y confiabilidad en la identificación de patrones visuales complejos, garantizando una detección precisa y confiable de señales de tránsito.

En el área académica, esta iniciativa no solo provocará un conjunto de oportunidades y beneficios experimentales, también influirá directamente en la formación académica del

estudiante en la universidad Estatal Península de Santa Elena, ya que se despertará la curiosidad del estudiante para encontrar maneras más adecuadas de implementación y modelos más complejos de detección de objetos dentro del área de la ingeniería.

En el área social, el proyecto servirá como una herramienta que reforzará la seguridad vial de la población mejorando la percepción visual del conductor mediante la visión por computadora. Sin embargo, el proyecto tendrá que pasar por una serie de pruebas para medir su rendimiento antes de ser implementado en un entorno real. Al mismo tiempo el sistema reforzará los estudios realizados en el país sobre el tema de detección de señales brindando nuevos resultados y mejorías en los algoritmos.

1.6. Alcance

El alcance de este proyecto abarca el desarrollo de un sistema de detección y reconocimiento de señales de tránsito mediante algoritmos de aprendizaje automático utilizando visión por computadora para ayudar a la percepción visual de conductor. El algoritmo de aprendizaje automático se escogerá de acuerdo a la investigación realizada sobre los métodos y técnicas utilizadas para el reconocimiento de imágenes u objetos, el algoritmo tendrá que pasar por una etapa de entrenamiento de acuerdo a una base de datos previamente adquirida, con la cual aprenderá a reconocer y clasificar las señales verticales de tránsito.

Debido a que la propuesta tecnológica está orientada a un entorno de pruebas de campo, el desarrollo del dispositivo se limita a la utilización de sistemas embebidos y módulos de fácil acceso.

1.7. Resultados Esperados

Dentro del desarrollo y entrenamiento del modelo utilizando entornos de programación se esperan los siguientes resultados acordes a los objetivos establecidos.

- Creación de una base de datos de entrenamiento amplio y diverso que refleje las variaciones reales del entorno de detección.
- Entrenamiento exitoso de un modelo de detección de objetos específicamente adaptado a las señales de tránsito en Ecuador, con especial atención a las señales de tránsito más importantes.
- Establecer valores de parámetros de entrenamiento como época, bacth y tamaño de imagen, de forma equilibrada para evitar el sobre entrenamiento del modelo.
- Obtener un rendimiento aceptable del 0.6 en mAP (precisión media) a umbrales de confianza de 50% y 95%

1.8. Metodología

En este apartado se describe la metodología empleada para el desarrollo de la propuesta. Además, se presenta aspectos fundamentales correspondientes al entrenamiento y ejecución de los modelos de detección escogidos.

1.8.1. Investigación Explorativa

Esta investigación se la realiza al inicio de un proyecto con el objetivo de estudiar y comprender el problema o tema a desarrollarse (Nelson Morales, 2015).

Se realiza la búsqueda de información en fuentes bibliográficas como tesis, proyectos o artículos científicos que ayuden a profundizar los siguientes temas: Redes neuronales artificiales, redes neuronales convolucionales, Deep learning, modelos de detección de objetos, entrenamiento.

1.8.2. Investigación Aplicada

Para este tipo de investigación, se aplicarán todos los conocimientos obtenidos en la investigación de bibliografías para el entrenamiento del modelo. Además, se utilizará entornos de programación para el entrenamiento del modelo de detección. Para la etapa final de la

propuesta se realizará la experimentación del sistema en un entorno de pruebas utilizando sistemas embebidos y módulos de fácil acceso para ver el funcionamiento y rendimiento del modelo entrenado.

1.8.3. Desarrollo de la propuesta

El desarrollo de la propuesta implica la ejecución de cuatro fases tomando como referencia el estudio (Blacio et al., 2021). En la primera fase, se lleva a cabo una revisión bibliográfica que abarca un análisis exhaustivo de los procesamientos de imágenes mediante inteligencia artificial y modelos de detección de objetos en imágenes (Montaño-Blacio et al., 2021). En la segunda etapa, se realiza un análisis de los modelos de detección de objetos más óptimos para el proceso de entrenamiento. En la tercera fase, se procede al entrenamiento de uno o más modelos seleccionados en la etapa anterior, con el objetivo de realizar una comparación de sus rendimientos en la detección de objetos. Finalmente, en la cuarta fase, se llevan a cabo pruebas con el modelo entrenado y análisis de los resultados obtenidos durante el entrenamiento.

1.8.3.1. Fase 1: Investigación bibliográfica

En esta fase, se realiza una revisión bibliográfica enfocada a procesamientos de imágenes con inteligencia artificial y modelos de detección de objetos. El objetivo principal es obtener un conocimiento detallado sobre las características y limitaciones de los modelos de detección propuestos en la investigación (Montaño et al., 2021).

La investigación se convierte en un punto clave ya que se obtiene características específicas de cada modelo. En el capítulo 2, se puede evidenciar los modelos de detección investigados y sus limitaciones.

1.8.3.2. Fase 2: Análisis del modelo más óptimo

En esta fase del proyecto, se lleva a cabo un análisis detallado con el objetivo de identificar el modelo de detección de objetos más adecuado para avanzar a la etapa de entrenamiento. Este análisis considera meticulosamente características clave, tales como la eficiencia computacional, los recursos requeridos y otros factores relevantes para asegurar la viabilidad y eficacia del modelo seleccionado.

1.8.3.3. Fase 3: Entrenamiento del modelo

En esta etapa clave del proyecto, se realiza el entrenamiento del modelo de detección, aprovechando herramientas de programación como Google Colab. Con una meticulosa atención a los detalles, se ajusta parámetros de entrenamiento perfeccionando la capacidad del modelo en la identificación precisa de las señales de tránsito.

1.8.3.4. Fase 4: Análisis del rendimiento del modelo en un entorno de pruebas

En esta etapa crucial, se evalúa meticulosamente el rendimiento del modelo previamente desarrollado. Se utiliza las curvas y métricas de pérdida, precisión y confianza para verificar si el modelo es eficiente o ineficiente para la detección de las señales de tránsito.

CAPÍTULO II

2. Fundamentación Teórica

En este apartado se realiza una revisión bibliográfica para cada uno de los temas importantes, esto permite analizar y definir conceptos que ayuden a comprender el funcionamiento y las características de la detección de objetos.

2.1. Señales de tránsito

Las señales de tránsito tienen su origen en la época del imperio Romano, fueron ellos los primeros en recurrir a la señalización de vías debido a los problemas que tenían cuando se trasladaban de un lugar a otro. Esta señalización se basaba en columnas de piedra y eran capaces de dar información a las personas dando a conocer ciertas características del camino, como el nombre, distancia que faltaba para llegar a la siguiente ciudad y donde desembocaba la vía. En el siglo XX las señales no presentaban modificaciones significativas, cada país creaba sus propias señales de acuerdo a la necesidad que presentaban. Sin embargo, con la llegada de los primeros automóviles se generaron diversos accidentes debido a la confusión de señaléticas, para ello se estableció una normativa de circulación. De esta forma nace el Congreso Internacional de Carreteras en el año de 1908 (Vial & Vial, 2015).

Las señales de tránsito son símbolos o signos ubicados en las vías públicas con el objetivo de transmitir información a los conductores que circulan por una vía o carretera. Cada señal representa un mensaje diferente y tienen el objetivo de controlar la circulación vehicular y reducir los accidentes de tránsito, además son unos de los aspectos más importantes en el área vehicular, ya que es el lenguaje de comunicación entre el conductor y las carreteras (Vial & Vial, 2015). A nivel mundial las señales de tránsito aplican el lenguaje de comunicación no verbal (símbolo), de esta manera, cualquier persona puede entenderlas sin importar el país en el que se encuentre. Las señales de tránsito logran salvaguardar la vida de la población, sin embargo, la

población en general debe cumplir y obedecer estas señales de lo contrario recibirán una sanción legal y lo que es peor, un accidente vehicular.

2.1.1. Señales verticales

Desde el año de 1968 las señales de tránsito se han regido dentro de un acuerdo a nivel global conocido como la convención de Viena, en el cual se decreta que todas señales de tránsito deben ser lo más idénticas posibles a nivel mundial. Las señales verticales están basadas en este principio, no obstante, cada país ha realizado pequeñas variaciones en la tipología de algunas señales, pero manteniendo en común las señales principales como el pare, ceda el paso y velocidad (Agua et al., 2014).

Las señales verticales son placas fijadas en estructuras como postes, los cuales se encuentran ubicados en las vías o carreteras, estas señales tienen la función de prevenir a los conductores sobre un peligro existente a unos cuantos metros del vehículo mediante una comunicación no verbal (símbolos y letras). Además de reglamentar las prohibiciones o restricciones vehiculares, estas señales brindan información a los usuarios que circulan por las calles sin vehículos con el objetivo de mantener la integridad de los mismos (Agua et al., 2014). De acuerdo con la función que cumplen, las señales verticales se pueden clasificar en:

- Señales preventivas
- Señales reglamentarias
- Señales informativas

2.1.1.1. Señales preventivas

Las señales preventivas son señales que alertan al conductor de peligros potenciales dando una descripción breve de su naturaleza, este peligro suele encontrarse a unos kilómetros del vehículo, el conductor debe tomar las debidas precauciones como detenerse, reducir la velocidad

o realizar una maniobra a fin de prevenir un accidente (Agua et al., 2014), la figura 1 indica algunas de estas señales que se ven a menudo en las calles y vías de circulación.

Algunas de estas señales verticales son:

- Señal lateral izquierda o derecha: Indica un empalme lateral de la vía en un ángulo aproximado a 90 grados.
- Bifurcación en “T”: Indica la proximidad de una bifurcación en T
- Bifurcación izquierda o derecha: Indica un empalme lateral de la vía en un ángulo menor a 60 grados.
- Superficie rizada: Alerta al conductor de una superficie con pavimento rizado, el conductor debe reducir la velocidad.
- Zona Escolar: Indica la proximidad a un cruce escolar.
- Peatones en la vía: Alerta al conductor de zonas donde hay movimiento de peatones.
- Curva pronunciada izquierda o derecha: Alerta al conductor de la presencia de curvas con un ángulo de deflexión es mayor a 80 grados.
- Curva izquierda o derecha: Alerta sobre curvas con un ángulo mayor a 20 y menor que 80 grados.
- Curvas sucesivas primero izquierda o derecha: se utiliza para representar 3 o más curvas.
- Curva o contractura a la derecha o la izquierda: Indica la presencia de dos curvas en sentido contrario cuyos ángulos son mayores a 20 grados y menores que 80 grados con una separación tangente de 120 metros.

De manera gráfica tenemos las señales verticales en la Figura 1.

Figura 1

Señales preventivas

Nota. Obtenido de (Agua et al., 2014)

2.1.1.2. Señales reglamentarias

Las señales reglamentarias tienen la función de informar a los conductores de las prioridades en el uso de las vías, a fin de regular el tránsito, además indican las limitaciones o prohibiciones impuestas por el gobierno (Agua et al., 2014). En la figura 2 se muestra las señales reglamentarias más comunes:

Figura 2

Señales reglamentarias

Nota. Obtenido de (Agua et al., 2014)

2.1.1.3. Señales informativas

Son señales de información vial cuyo objetivo es orientar y guiar a los conductores mediante imágenes que indiquen como llegar a un lugar de la forma más simple, segura y directa (Agua et al., 2014). La información que brindan estas señales son las direcciones, sitios de interés, localidades, distancias, intersecciones como se indica en la figura 3.

Figura 3

Señales informativas

Nota. Obtenido de (Agua et al., 2014)

2.2. Procesamiento de imágenes mediante inteligencia artificial

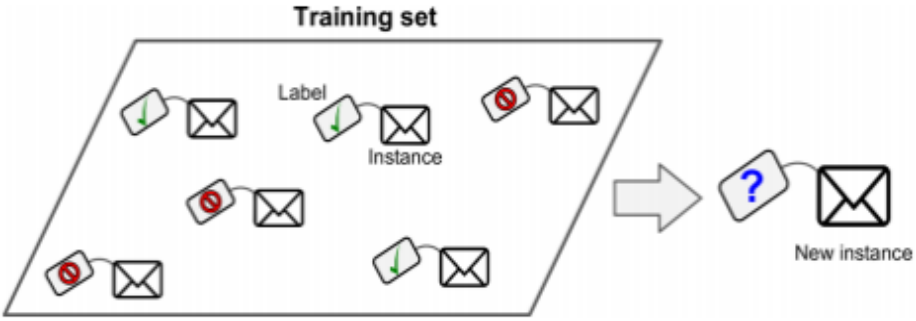
El procesamiento de imágenes mediante Machine Learning consiste en utilizar algoritmos de aprendizaje automático para automatizar y mejorar el análisis de patrones complejos en conjuntos de datos o patrones extraídos de una imagen. A diferencia de la programación clásica, en la que una persona ejecuta el código varias veces de forma manual con el objetivo de extraer datos, Machine Learning realiza esta tarea de forma autónoma. Machine Learning permite a las máquinas aprender patrones y características por sí mismas a partir de datos pasados obtenida del usuario o una base de datos. La gran variedad de algoritmos destinados al aprendizaje

autónomo ha llevado a clasificarlos según el tipo de supervisión y de cuanta supervisión necesitan para su eficiencia, dicho esto, se tiene 4 categorías (González-García):

- **Aprendizaje supervisado:** es un enfoque común en el procesamiento de imágenes mediante machine learning, en el que se alimenta al algoritmo con una gran cantidad de datos de imágenes etiquetadas con información de la clase a la que pertenece cada imagen. El algoritmo utiliza estos datos de entrenamiento para aprender a clasificar las imágenes en diferentes categorías ("Sanchez Vidal," 2021). Un ejemplo común del aprendizaje supervisado es la clasificación de los correos spam y no spam como se muestra en la figura 4.

Figura 4

Aprendizaje supervisado

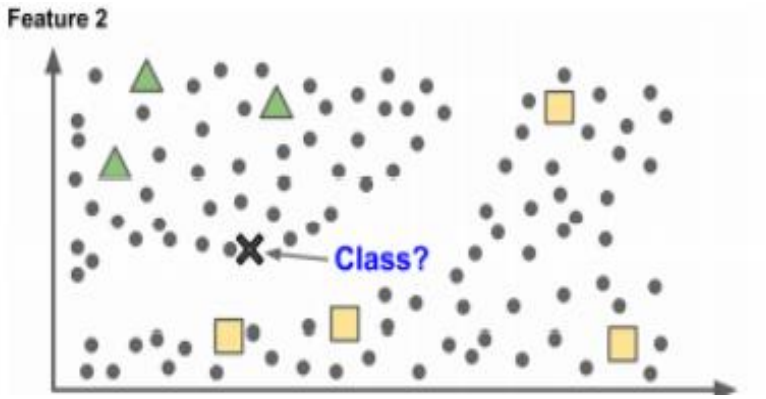


Nota. Obtenido de ("Sanchez Vidal," 2021)

- **Aprendizaje semisupervisado:** en este aprendizaje, los datos están parcialmente etiquetados, es decir, una parte tendrá etiquetas mientras que la otra no. En la figura 5 se observa dos grupos, uno con etiquetas y otro sin etiqueta, siendo mayor cantidad los datos que no se encuentran etiquetados ("Sanchez Vidal," 2021).

Figura 5

Aprendizaje semisupervisado



Nota. Obtenido de (“Sanchez Vidal,” 2021)

- **Aprendizaje por refuerzo:** en este aprendizaje se utiliza un sistema alterno conocido como “agente”, este deberá aprender a tomar decisiones mediante la prueba y el error en donde deberá a escoger la mejor decisión de acuerdo a las recompensa o penalización (“Sanchez Vidal,” 2021), como se observa en la figura 6.

Figura 6

Aprendizaje por refuerzo

Nota. Obtenido de (“Sanchez Vidal,” 2021)

El procesamiento de imágenes mediante machine learning tiene una amplia variedad de aplicaciones prácticas, como el reconocimiento de objetos, la detección de fraudes en la banca,

la detección de enfermedades médicas y la clasificación de imágenes satelitales. A medida que la tecnología continúa evolucionando, es probable que veamos aún más avances en este campo.

2.3. Deep Learning

Dentro de la inteligencia artificial denominada machine learning se encuentra una técnica basada en las conexiones del cerebro humano con el cual una máquina puede aprender automáticamente funciones y desempeñarlas en diferentes áreas según la necesidad. En el área del procesamiento de imágenes existe un problema llamado segmentación, este problema puede ser tratado mediante los algoritmos que brinda el Deep Learning, de hecho, debido a que es una tecnología de aprendizaje automático, esta puede ser de manera supervisada y no supervisada (Jhan & Arteaga, 2015).

Deep Learning ha demostrado ser una herramienta factible y capaz para resolver problemas visuales como la visión por computadora y otro procesamiento de imágenes como:

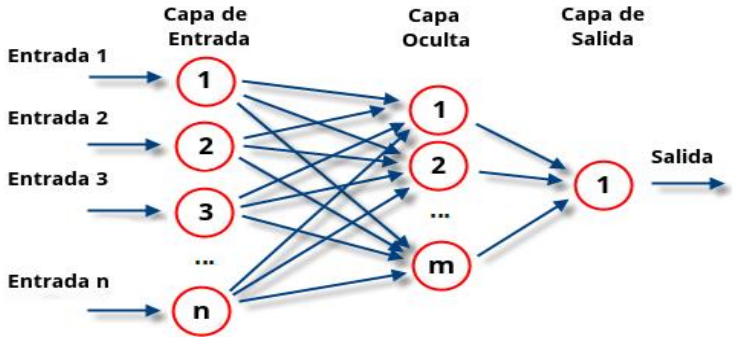
- **Clasificación de imágenes:** El Deep learning es una herramienta efectiva para la clasificación de imágenes de acuerdo a las etiquetas que se coloca a cada imagen en la base de datos la cual va utilizar el algoritmo para el entrenamiento del sistema
- **Detección de objetos:** Esta herramienta nos brinda la posibilidad de reconocer patrones dentro de una imagen limitada por un cuadro de interés, debido a esto los objetos que se encuentran dentro de este cuadro pasan a ser reconocidos por el algoritmo creando una base de datos dentro del sistema.
- **Reconocimiento facial**: se puede entrenar al sistema para reconocer rostros, esto es similar al reconocimiento de objetos, sin embargo, las imágenes deben ser etiquetadas y más extensa debido a las diferentes variaciones que pueden existir dentro de un lugar.

2.4. Redes neuronales artificiales

Las redes neuronales constituyen una categoría de algoritmo de aprendizaje automático que se inspira en la estructura y operación del cerebro humano. Estas redes consisten en un conjunto de nodos interconectados que aprenden de datos pasados con los cuales realizan cálculos y envían señales a otras neuronas con el objetivo de realizar una predicción como se indica en la figura 7. Las redes neuronales artificiales son modelos estadísticos no lineales que muestran una relación compleja en sus entradas y salidas a fin de descubrir un nuevo patrón (López & Fernández, 2008).

Figura 7

Redes neuronales artificiales



Nota. Obtenido de (López & Fernández, 2008)

Existen diferentes tipos de redes neuronales artificiales con diferentes componentes. Sin embargo, los componentes más importantes que se encuentran dentro de todas las redes neuronales son:

1. Neuronas

Se dividen en tres grupos:

- Neuronas de entrada: reciben información del mundo exterior

- Neuronas ocultas: procesan la información obtenida por las neuronas de entrada
- Neuronas de salida: producen una predicción

2. Pesos

Los pesos se suman a los cambios de información de entrada. Los resultados de la neurona con mayor peso serán dominantes y se transmiten a la siguiente neurona, mientras que la información de la neurona con menor peso no se transmitirá.

3. Bias

Una neurona de bias permite almacenar más variaciones de pesos. Para las redes neuronales artificiales se necesita una neurona de bias en cada capa, esta desempeña el proceso de desplazamiento de la función de activación hacia la izquierda o a la derecha en la red.

2.5. Redes neuronales convolucionales

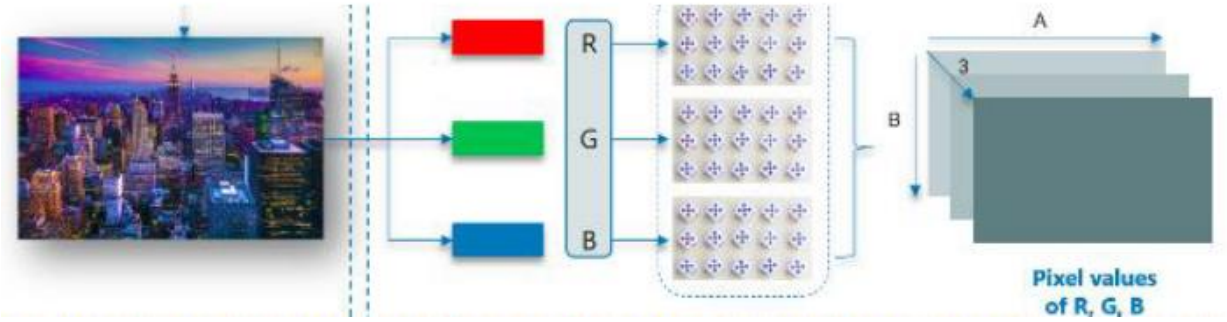
Una red neuronal convolucional es un sistema que consta de múltiples capas convolucionales alternadas que permiten realizar diversos procesos en donde cada capa tiene un proceso diferente y único. Además, las capas convolucionales tienen k filtros con dimensiones que generalmente son $n \times n \times q$, en donde n y q suelen ser características escogidas por el diseñador de la red, sin embargo, cuando se trabaja con redes convolucionales se debe tener en cuenta que cada capa realiza un proceso diferente, esto quiere decir que cada capa genera nuevos resultados provocando que el sistema se demore un cierto tiempo para llegar al resultado. Si la imagen del sistema es muy grande en sus pixeles, puede provocar un gran retraso en el proceso (Durán Suárez et al., 2017).

La figura 8 muestra el ejemplo de Artola Moreno, Á. (2019). En este ejemplo se pueden observar una imagen de 50 x 50 pixeles de alto y ancho, esto quiere decir que tendremos 2500 neuronas en el sistema. En este caso la imagen es a color, debido a esto tenemos tres canales:

red, green y blue, utilizando un total de 50 x 50 x 3 = 7500 neuronas en la entrada. Tras realizar los procesos de normalización de valores se comienza con los procesos de las convoluciones lo cual consiste en tomar un grupo de pixeles cercanos de la entrada e ir procesando matemáticamente contra una matriz llamada kernel conocida como filtro. Este filtro recoge la matriz de entrada de arriba hacia abajo y de izquierda a derecha, dando como resultado una nueva matriz de salida denominada matriz de activación, de esta manera el sistema procesa la imagen y una vez realizada la convolución mediante la matriz de activación, nos devuelve una salida a partir de un valor de entrada que generalmente se encuentra en un rango de (0,1) o (-1,1) (Artola Moreno Tutor & Antonio Pérez Carrasco, 2019).

Figura 8

Redes neuronales convolucionales



Nota. Obtenido de (López & Fernández, 2008)

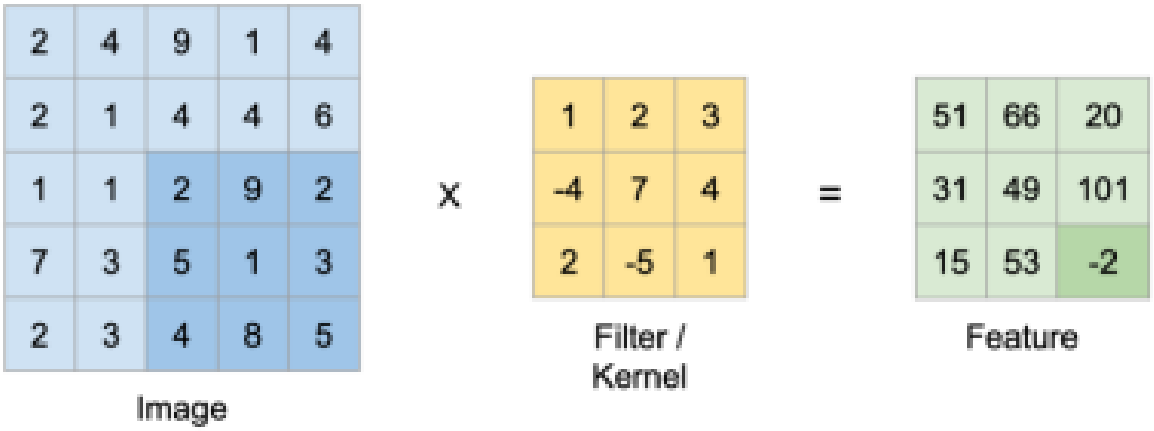
2.5.1. Capas convolucionales

Las capas convolucionales son un tipo de capa de las redes neuronales convolucionales que se utilizan para procesar datos que se tiene en una estructura espacial como imágenes, videos o audios. En estas capas se aplican varios filtros (también denominados kernel) a la entrada de la capa, estos filtros recorren la imagen de arriba abajo y de izquierda a derecha dando como resultado un “mapa de características” que resalta una de las varias características de la imagen.

Cada filtro produce un mapa de datos que resaltan características específicas como bordes, textura, color, etc. Estas características se unen para tener una imagen más abstracta con el cual se pueden seguir haciendo los siguientes procesos. Además, estos mapas de características trabajan con ejes como altura, anchura y canales, la figura 9 muestra una imagen de 28x28 en escala de grises a la cual se le aplica 32 filtros, en esta imagen se obtendrán 32 matrices de salida de 28x28x1 cada una, esto provoca una red neuronal de 25.08, esta cifra es muy elevada para una imagen de 28x28. Para reducir estos números de neuronas tan elevada aparecen las capas pooling (“Sanchez Vidal,” 2021).

Figura 9

Capas convolucionales



Nota. Obtenido de (“Sanchez Vidal,” 2021)

2.5.2. Capas Pooling

Estas capas son muy importantes y suelen colocarse después de las capas convolucionales debido a que reducen las dimensiones de salidas resultantes manteniendo las características más relevantes detectadas que son producidas en el proceso de convolución. Cuando se realiza el proceso de submuestreo se escoge de un grupo de $n\ x\ m$ pixeles uno solo, según los diferentes criterios que se usa para la elección del pixel los cuales son:

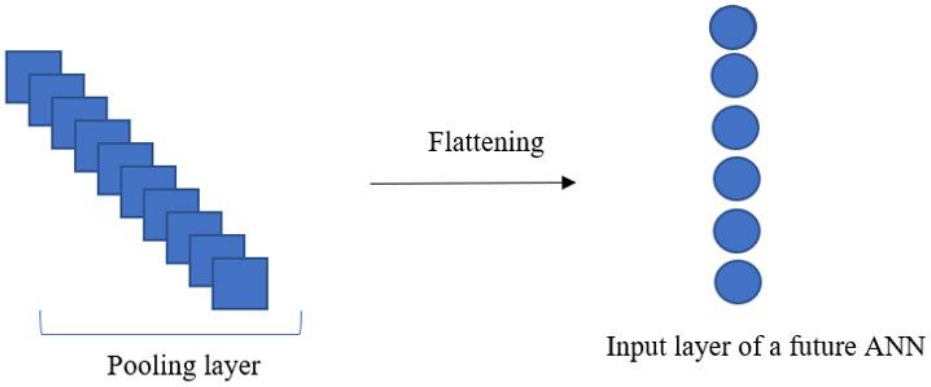
- Max-Pooling: se queda con el pixel de mayor valor
- Min-Pooling: se queda con el pixel de menor valor
- Average-Pooling: hace la media de los $n\ x\ m$ pixeles

2.5.3. Capas flatten

Se utiliza para transformar los datos de entrada bidimensionales en un formato unidimensional antes de ser conectados a una o más capas como se muestra en la figura 10.

Figura 10

Capa flatten



Nota. Obtenida de (“Sanchez Vidal,” 2021).

2.5.4. Capas completamente conectadas

Mediante los procedimientos de las capas pooling, se genera un vector cuyo objetivo es servir como capa de entrada de una red neuronal. Una vez recorrido todas las neuronas de las diferentes capas se llega la última capa denominada capa de salida, esta capa es la encargada de clasificar la imagen de entrada y asignarle una clase. Hay procesos en los cuales se necesitan hacer múltiples clasificaciones, para ello, se utiliza la función Softmax que consiste en tener el mismo número de neuronas y clases, de esta forma, se asignará a cada clase una probabilidad decimal en un caso múltiple cuya suma debe dar como resultado 1 (“Sanchez Vidal,” 2021).

2.6. Modelos de reconocimiento de objetos en imágenes

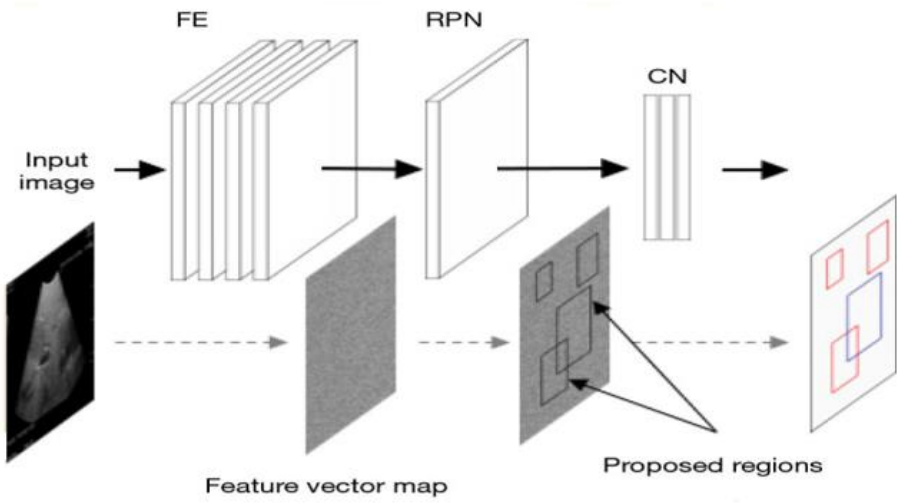
Los modelos de reconocimientos de objetos en imágenes son algoritmos de aprendizaje automático que aplica la inteligencia artificial con el objetivo de detectar y clasificar objetos dentro de las imágenes tomando en consideración características específicas del objeto o llevando una secuencia de patrones dentro de la imagen. Estos sistemas de reconocimientos se aplican a diversas áreas de trabajo, desde la seguridad por medio de cámaras de video vigilancia hasta diagnósticos médicos (Gómez Flores, 2015).

2.6.1. Faster R-CNN

Es un método de detección de objetos que consta de tres componentes: un extractor de características (FE), RPN y una red neuronal (CN). Estas tres capas tienen una red convolucional de extremo a extremo, lo que permite un procesamiento de información más rápido como se muestra en la figura 11 (Cerpas Alonso, 2021).

Figura 11

Faster R-CNN



Nota. Obtenida de (Cerpas Alonso, 2021).

El extractor de características (FE) corresponde a una red neuronal convolucional profundamente conectada que al presentarse una imagen como entrada, produce un conjunto de

vectores de características que representan los objetos presentes en la figura 11. Estos vectores son expresados mediante un arreglo multidimensional que captura la información cercana a cada pixel del objeto. Además, para que el extractor de características (FE) obtenga vectores requiere se entrenada por un conjunto grande de imágenes.

RPN se refiere a una red convolucional encargada de identificar las áreas que contienen objetos en una imagen, utilizando un mapa vectorial de características generado por FE. Por otro lado, CN es una red neuronal conectada que realiza la predicción tanto de la etiqueta de un objeto como el área específica donde los objetos pueden ubicarse. Estas predicciones se basan en las regiones propuestas y en el mapa vectorial de características (Cerpas Alonso, 2021).

A pesar de ser un modelo sólido para detección de objetos tiene algunas limitaciones que se deben tener en cuenta:

- **Velocidad de inferencia limitada:** a pesar de ser una mejora a otros módelos, Fast R-NN todavía puede ser lento. La necesidad de realizar múltiples pasadas a través de la red puede afectar la velocidad de inferencia.
- **Proceso de entrenamiento secuencial:** el proceso de entrenamiento se lleva a cabo en dos etapas separadas, una para la generación de propuestas y otra para la clasificación y regresión
- **Recursos de computación significativos:** Fast R-NN requiere recursos computacionales mas significativos, especialmente al usar redes troncales mas grandes como VGG16.
- **Manejo de objetos pequeños:** puede presentar dificultades al detectar objetos pequeños debido a las dependencias de regiones de interés generadas externamente.

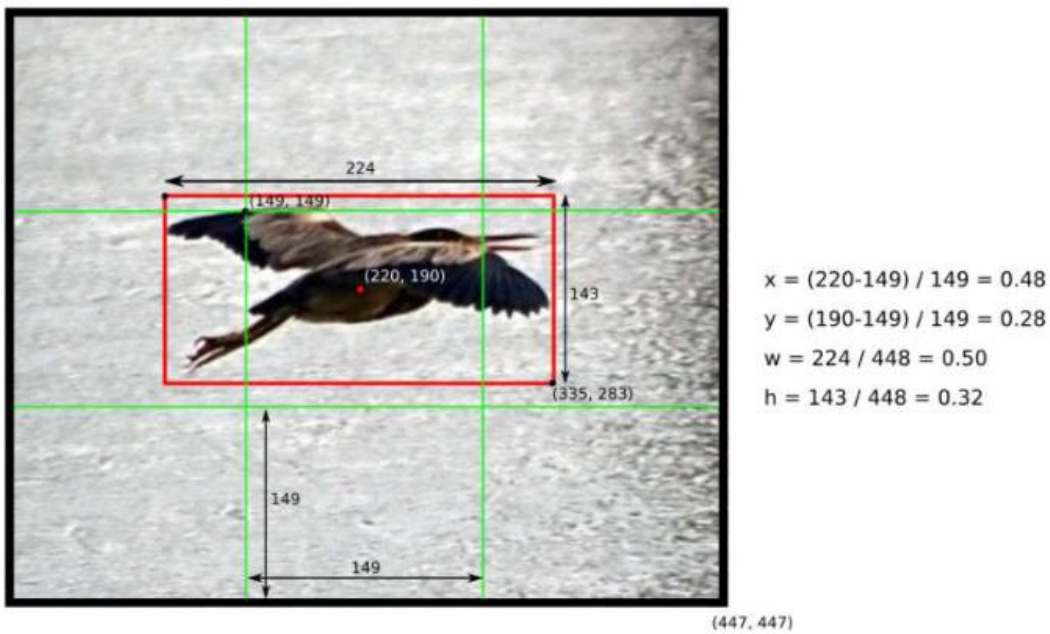
2.6.2. Yolo (You Only Look Once)

Es una arquitectura de detección de objetos rápida, precisa y factible, fue creada por Joseph Redmond. Esta arquitectura es ideal para la detección de objetos en imágenes o video en tiempo real debido a sus cualidades de rápida detección. YOLO es un algoritmo de aprendizaje profundo que utiliza redes neuronales convolucionales para detectar y clasificar objetos en una imagen mediante cuadros delimitadores (Saúl & Raneros, 2021).

Yolo se fundamenta en una red neuronal convolucional que emplea las características de una imagen para identificar y dibujar cuadros limitadores alrededor del objeto de interés. Esta imagen se divide en rejillas de tamaño S x S de modo que, si un objeto cae dentro de una celda de la cuadrícula, esa celda es la responsable de reconocer y detectar ese objeto como se puede observar en la figura 12.

Figura 12

Funcionamiento de Yolo



Nota. Obtenida de (Saúl & Raneros, 2021).

Yolo es un modelo detección de objetos eficiente y popular, a pesar de ello presenta algunas limitaciones:

- **Detección de objetos adyacentes:** Yolo tiende a tener problemas pequeños a su enfoque de muestreo de la imagen en celdas de cuadrícula, los objetos no pueden llegar a ser representados adecuadamente dentro de la cuadrícula.

- **Sensibilidad a la distribución de la cuadrícula:** para los objetos que se encuentran predominante en una región específica, la cuadrícula puede no ser lo suficientemente fina para capturar los detalles.

- **Escenarios de densidad variable:** las detecciones de objetos en imágenes con una densidad variable de objetos puede ser un desafío, ya que la red puede no adaptarse a la variabilidad en la distribución de objetos.

- **Recurso de computación significativos:** los modelos más actuales y más pesados requieren de recursos computacionales más potentes.

2.6.3. Teachale Machine

Teachable Machine es una plataforma en línea desarrollada por Google que permite a las personas, incluso aquellas sin experiencia en programación o aprendizaje automático, crear modelos de aprendizaje automático de manera sencilla y rápida. La plataforma está diseñada para ser accesible y amigable para usuarios no técnicos, lo que la convierte en una herramienta educativa y creativa que permite a las personas explorar conceptos de aprendizaje automático de forma práctica (Carolina Armendariz Rodríguez et al., 2023).

A través de Teachable Machine, los usuarios pueden realizar las siguientes acciones:

- **Entrenamiento de Modelos:** Los usuarios pueden cargar imágenes, audio o datos de sensores para entrenar modelos de aprendizaje automático. La plataforma utiliza una técnica conocida como transfer learning, que aprovecha modelos de aprendizaje automático preentrenados para facilitar el proceso de entrenamiento.

- **Clasificación de Datos:** Una vez que se ha entrenado un modelo, los usuarios pueden utilizarlo para clasificar nuevas muestras de datos. Por ejemplo, pueden entrenar un modelo para reconocer diferentes objetos en imágenes y luego usar ese modelo para clasificar nuevas imágenes.

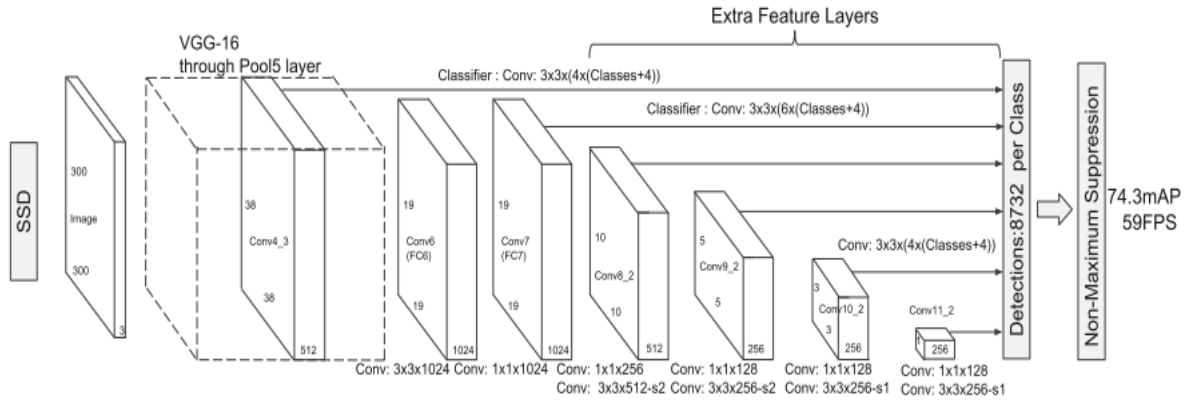
- **Exportación de Modelos**: Los modelos entrenados en Teachable Machine se pueden exportar para su uso en aplicaciones web, aplicaciones móviles o proyectos personales.

2.6.4. Detector de cajas múltiples de único disparo (SSD)

El modelo de SSD se fundamenta en una red neuronal convolucional progresiva que genera un conjunto de cuadros delimitadores y puntuaciones de manera constante para identificar la presencia de instancias de clases de objetos en esos cuadros. Posteriormente, se lleva a cabo un proceso de supresión de no máximos, para obtener las detecciones definitivas. Las capas iniciales de la red siguen una arquitectura estándar empleada en la clasificación de imágenes de alta calidad, a la que denominamos red de base (Liu et al., 2016). A continuación, en la figura 13 se puede observar la estructura adicional en la red para generar detecciones que posean las siguientes características clave:

Figura 13

Estructura de SSD



Nota. Obtenida de (W. Liu et al., 2016).

SDD es un modelo eficiente para la detección de objetos, pero también presenta algunas limitaciones:

- **Problemas con superposición de objetos:** presenta problemas en la detección de objetos que se encuentren muy cercanos, ya que su enfoque es en cuadrículas de detección, esto provoca dificultades para asignar cuadros delimitadores precisos.
- **Mayor número de falsos positivos:** puede tener un mayor número de falsos positivos, especialmente en escenarios con objetos pequeños.
- **Menor precisión en la clasificación**: la clasificación puede ser ligeramente inferior, ya que la red esta diseñada para optimizar la eficiencia y la velocidad de detección.
- **Objetos de forma irregular**: los cuadros delimitadores pueden no adaptarse bien a estas formas.

2.6.5. RenitaNet

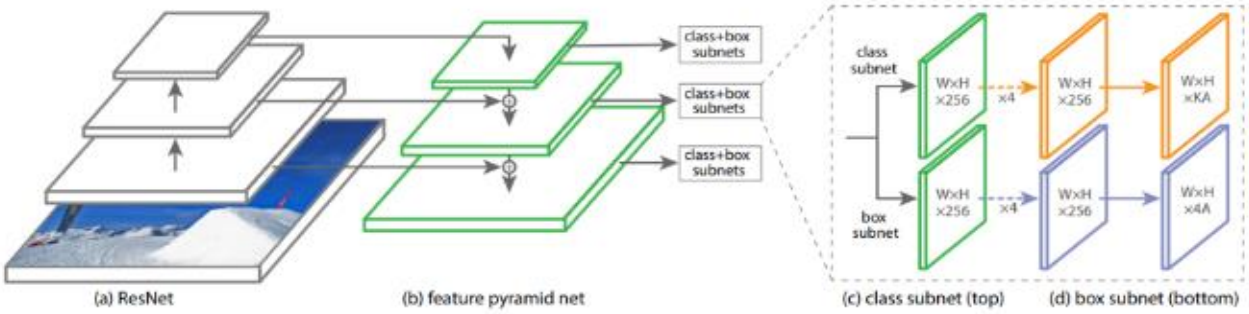
RetinaNet es un modelo de detección de objetos que utiliza una sola red unificada que consta de una red troncal y dos subredes especializadas como se obseva en la figura 14. La red troncal se encarga de calcular un mapa de características convolucionales para toda la imagen de entrada. Mientras que las subredes, son diseñadas de manera simple para facilitar una detección densa en una sola etapa, en donde se realizan la clasificación de objetos y la regresión de los cuadros delimitadores.

RetinaNet aborda el desequilibrio de clases durante el entrenamiento mediante el uso de una función de pérdida focal. Esta función modúla la pérdida de entropía cruzada, centrándose en ejemplos negativos específicos y escalando dinámicamente la pérdida. La escala disminuye a medida que la confianza en la clase correcta aumenta, reduciéndola influencia de ejemplos

fáciles y enfocando el modelo en casos mas desafiantes. La pérdida focal introduce un factor en el criterio de entropía cruzada estándar, donde un valor disminuye la pérdida relativa de ejemplos bien clasificados, priorizando ejemplos difíciles y mal clasificados. Este enfoque es crucial para detectores de objetos candidatos en una imagen (Li et al., 2021).

Figura 14

Redes troncales de RentinaNet



Nota. Obtenida de (Li et al., 2021)

CAPÍTULO lll

3. Modelos y herramientas de la propuesta

3.1. Regresión y clasificación basada en marcos delimitadores

Los modelos basados en cuadros regionales delimitadores, están compuestos por varias etapas correlacionadas como: extracción de características con CCN, clasificación y cajas delimitadoras.

3.1.1. YOLO

Yolo se basa en una red convencional única, que predice múltiples cuadros delimitadores y las probabilidades asociadas a las clases correspondientes. En comparación con otros métodos, YOLO se destaca por su velocidad; la primera versión procesada en una GPU Titan X logra aproximadamente 45 fps, mientras que una versión más rápida alcanza los 150 fps. Este rendimiento implica la capacidad de procesar transmisiones de video en tiempo real con una latencia inferior a 25 milisegundos. Es importante mencionar que todo el código de entrenamiento y prueba es de código abierto, y hay varios modelos preentrenados disponibles para su descarga (Ameijeiras Sánchez & Hernández Heredia, 2020).

La estructura de la red de Yolo se fundamente en Google Net, diseñado originalmente para clasificación de imágenes. Consta de 24 capas convolucionales consecutivas, seguidas por 2 capas completamente conectadas. Se emplean capas de resolución de dimensiones 1x1, seguidas de capas convolucionales 3 x 3. Yolo implementa la predicción de múltiples cajas delimitadoras por cada celda de la cuadrícula durante el entrenamiento. Cada objeto es asignado a un predictor de caja según la mayor similitud con la zona correcta, basada en el valor de intersección sobre unión (IoU). Este enfoque especializado facilita la mejora de la predicción en términos de tamaños, relaciones de aspecto y clase de objetos, lo que contribuye al rendimiento general del modelo (Ameijeiras Sánchez & Hernández Heredia, 2020).

Existen versiones de Yolo con características y cargas computacionales diferentes. A continuación, se nombra una versión media y alta a fin de hacer comparaciones.

3.1.1.1. YOLOv5

Se trata de un sistema de detección de objetos en tiempo real de código abierto, es decir, el modelo está capacitado para ser entrenado con cualquier base de datos. Dentro de Yolov5 se tiene modelos con características y recursos computacionales diferentes, la tabla 1 muestra las características de cada uno de ellos (Purwita Sary et al., 2023).

Tabla 1

Versiones Yolov5n

Modelo	Tamaño (pixeles)	mAP 50-95	Velocidad CPU	Parámetros (M)
Yolov5n	640	28.0	45	1.9
Yolov5s	640	37.4	98	7.2
Yolov5m	640	45.4	224	21.2
Yolv5l	640	49.0	430	46.5
Yol5x	640	50.7	766	86.7

Nota. Tabla modifica de la pagina oficial de yolov5 Ultralytics

3.1.1.2. YOLOv8

YOLOv8 es la última versión de Yolo en donde no solo se realiza una detección, sino que también se utiliza técnicas de segmentación en tiempo real. La tabla 2 muestra características claves de estos modelos (Purwita Sary et al., 2023).

Tabla 2 *Versiones de Yolov8x*

Modelo	Tamaño (pixeles)	mAP 50-95	Velocidad CPU	Parámetros (M)
Yolov8n	640	37.3	80.4	3.2
Yolov8s	640	44.9	128.4	11.2
Yolov8m	640	50.2	234.7	25.9
Yolv8l	640	52.9	375.2	43.7
Yol8x	640	53.9	479.1	68.2

Nota. Tabla modifica de la pagina oficial de yolov8 Ultralytics

3.1.1.3. Dependencias de Yolo

Para utilizar Yolo, generalmente se necesitará instalar cierta dependencias y bibliotecas. Teniendo cuenta que estos requisitos pueden cambiar con nuevas versiones. A continuación, la tabla 3 proporciona algunas de estas dependencias (Luis M García., 2021):

Tabla 3

Dependencias de Yolo

Librería	Descripción	Descripción
Matplotlib	Módulo de Python que contiene librerías necesarias para crear visualizaciones.	Se utiliza para representar en graficas el rendimiento del modelo según su etapa.
Numpy	Librería que da soporte para todas las herramientas matemáticas.	Se utiliza para arreglos multidimensional, funciones matemáticas de alto rendimiento y

		herramientas para trabajar con estos arreglos.
Torch	Biblioteca de aprendizaje profundo creada por Facebook	Proporciona un interfaz dinámica y flexible que facilita la experimentación con modelos y prototipos
Tensor Board	Conjunto de librerías con la mantendremos un seguimiento al proceso de aprendizaje de gráficos.	Se utiliza para monitorear métricas importantes, como la precisión y la pérdida, a medida que el modelo se entrena.
Seaborn	Biblioteca de visualización de datos en Python basada en matplotlib.	Proporciona una interfaz de alto nivel para crear gráficos estadísticos.
OpenCV	Librería de Python esencial para procesamiento de imágenes.	Se utiliza para manipular imágenes y videos donde se aplican técnicas de detección de objetos.

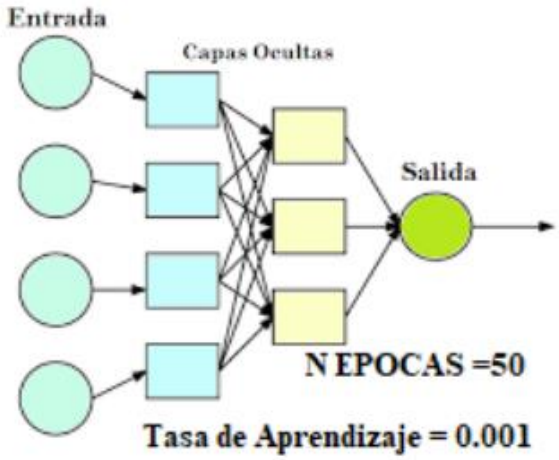
Nota. Tabla btenida de (Luis M García., 2021).

3.2. Teachale Machine

Tachable Machine es una plataforma en línea gratuita de Google que permite a usuarios sin experiencia en programación crear modelos de aprendizaje automático. Con un enfoque en el aprendizaje profundo, especialmente mediante Redes Neuronales Artificiales, la herramienta facilita el entrenamiento eficiente de modelos para reconocer imágenes, sonidos y pesos en tiempo real. Teachable Machine tiene la opción de exportar los modelos para ser utilizados en aplicaciones web o móviles. Su simplicidad en el uso ha atraído a individuos de diversas edades y niveles de destreza, quienes la emplean en el desarrollo de proyectos creativos, educativos o comerciales. La figura 15 ilustra la arquitectura de la red neuronal (Carolina Armendariz Rodríguez et al., 2023).

Figura 15

Arquitectura de Teachable Machine



Nota. Obtenida de (Carolina Armendariz Rodríguez et al., 2023).

Para medir la precisión de la Red Neuronal se utiliza la matriz de confusión donde se encuentran los siguientes aspectos:

- **Verdaderos positivos (VP):** se refiere a los casos en que el modelo ha realizado una predicción correcta al identificar correctamente que una muestra pertenece a una clase específica.
- **Falso positivo (FP):** se refiere a los casos que el modelo hace una predicción incorrecta al identificar erróneamente que una muestra pertenece a la clase negativa en lugar de la positiva.
- **Falsos negativos (FN):** Ocurre cuando el modelo predice un valor negativo, pero en realidad es positivo, esto ocurre porque no logra detectar correctamente el objeto.
- **Verdadero negativo (VN):** se refiere a la situación en el cual el modelo logra reconocer acertadamente un objeto incluso cuando dicho objeto no está realmente presente la imagen.

Teachable Machine es una herramienta que permite crear redes neuronales con facilidad. Sin embargo, tiene dificultades en su reconocimiento y limitaciones en cuanto a reconocimientos de patrones, ya que depende de factores como la iluminación y el fondo utilizado en el modelo. A diferencia de herramientas como OpenCV y TensorFlow, Teachable Machine no cuenta con la capacidad de analizar detalladamente patrones.

3.3. Configuración de parámetros de entrenamiento

Los parámetros de entrenamiento son una parte fundamental, ya que con ello podemos optimizar nuestro modelo, estos parámetros se configuran de acuerdo a las cargas computacionales y el software de programación. Existen varias herramientas de programación que son utilizados para desarrollar modelos en diferentes lenguajes y para diversas áreas de trabajo. La elección de la herramienta va depender del lenguaje de programación que desea utilizar, así como del proyecto que se esté realizando. La tabla 4 muestra las herramientas de programación que se han seleccionado para este proyecto.

Tabla 4

Herramientas de programación

Entorno	Descripción
PyCharm	Un IDE para Python que ofrece funciones como autocompletado de código, depuración y soporte para frameworks.
Google Colab	Un entorno de cuadernos basados en la nube que permite ejecutar códigos de Python en servidores de Google

3.3.1. Épocas

Las épocas indican el número de veces que nuestro modelo va recorrer todo el conjunto de datos durante todo el proceso de entrenamiento. Este parámetro puede variar según el modelo ya que un modelo con baja precisión va necesitar más épocas que un modelo de mayor precisión. Sin embargo, para encontrar el número exacto de épocas para entrenar el modelo es recomendable seguir las estadísticas y resultados del entrenamiento, ya que un modelo con un número excesivo de épocas puede conllevar a un sobre entrenamiento, haciendo que la red detecte objetos que sean iguales a los de la base de datos (Rosa Gonzales et al., 2022).

3.3.2. Batch

El parámetro “batch” se refiere al tamaño del lote o al número de imágenes utilizadas en cada paso de entrenamiento. Por ejemplo, si se tiene un conjunto de 2000 imágenes y se establece el parámetro “batch” en 32, se utilizarán 32 imágenes en cada paso de entrenamiento. Esto indica que, antes de seguir al siguiente lote, el modelo actualizará sus pesos y realizará los cálculos necesarios en función de 32 imágenes (Rosa Gonzales et al., 2022).

3.3.3. Tamaño de imágenes

La resolución de las imágenes es un parámetro fundamental en el proceso de entrenamiento, pues una imagen cuanto más pixel tenga, mejor calidad tendrá, pero mayor recurso necesitará el modelo para trabajar con ella. En Yolo, se maneja un valor de 640 pixeles en el lado más largo de la imagen, si se trabaja con una imagen de 640x480, el lado más largo correspondiente a su ancho, sería de 640 pixeles (Rosa Gonzales et al., 2022).

3.4. Herramientas de evaluación del modelo

Existen una variedad de herramientas y métricas específicas que pueden ser utilizadas para evaluar el rendimiento del modelo de detección. A continuación, se presenta algunas herramientas y métricas más comunes.

3.4.1. Presición promedio media (mAP)

Es una métrica utilizada para medir el rendimiento del modelo entrenado. Esta métrica toma a consideración la precisión promedio en diferentes niveles de sensibilidad (recall), donde se calcula la precisión promedio de cada clase y luego se toma el promedio general. En el contexto de modelos de detección de objetos, existen variantes de esta métrica, por ejemplo:

- **mAP-50**: evalúa la precisión promedio considerando un umbral de intersección del 50 %. Este umbral indica la cantidad de superposición requerida entre la caja delimitadora de predicción y la caja delimitadora real.
- **mAP50-95**: evalúa la precisión promedio del modelo tomando en consideración colores de intersección dentro de un grado del 50% hasta el 95%. Esta métrica implica calcular la precisión promedio de cada clase y nuevamente tomar el promedio de todos esos valores.

3.4.2. Sensibilidad (recall)

Sensibilidad o tasa de verdaderos positivos, es una métrica de evaluación que representa la proporción de instancias positivas que fueron correctamente identificadas en relación con instancias positivas reales. A continuación, se presenta la ecuación 1:

Ecuación 1

Recall

$$Recall = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$

Donde:

- **Verdaderos positivos (TP):** son los casos donde el modelo predice correctamente una instancia positiva.

- **Falsos negativos (FN):** son los casos donde el modelo no predijo correctamente una instancia.

3.4.3. Confianza

Se asocia con la puntuación de confianza asignadas a cada predicción. En modelos de detección rápida, cada caja delimitadora de predicción tiene una puntuación de confianza que indica cuan seguro está el modelo en esa predicción. A menudo, la puntuación de confianza se normaliza entre 0 y 1.

3.4.4. Matriz de confusión

La matriz de confusión es una herramienta que se utiliza para evaluar el modelo de detección, en esta matriz se comparan las predicciones de un modelo con las clases reales de los datos. La matriz clasifica las predicciones de la siguiente manera:

- **Verdaderos positivos (VP):** casos en que el modelo predice correctamente que la instancia pertenece a una clase específica.
- **Verdaderos negativos (VN):** casos en donde modelo predice correctamente que la instancia no pertenece a una clase.
- **Falsos positivos (FP):** casos donde el modelo predice incorrectamente que la instancia pertenece a una clase cuando no lo es correcto.
- **Falsos negativos (FN):** casos que el modelo predice incorrectamente que una instancia no pertenece a una clase cuando si lo es.

3.4.5. Métricas de pérdida

En el contexto de Yolo, las métricas de pérdida se refieren a diferentes componentes de la función del de pérdida del modelo. Estas funciones de pérdida se las divide en las siguientes:

- **Box_loss (pérdida de caja):** es la métrica que mide las predicciones de las coordenadas de la caja delimitadora que rodean el objeto a identificar con las cajas delimitadoras reales.
- **Obj_loss (pérdida de objetividad):** es la metica que mide la capacidad del modelo que tiene para predecir si hay un objeto presente en una determinada región de la imagen.
- **Cls_loss (pérdida de clasificación):** mide la precisión en la predicción de cada clase, cuando el modelo predice instancias negativas como positivas entonces la pérdida cls_loss aumenta.

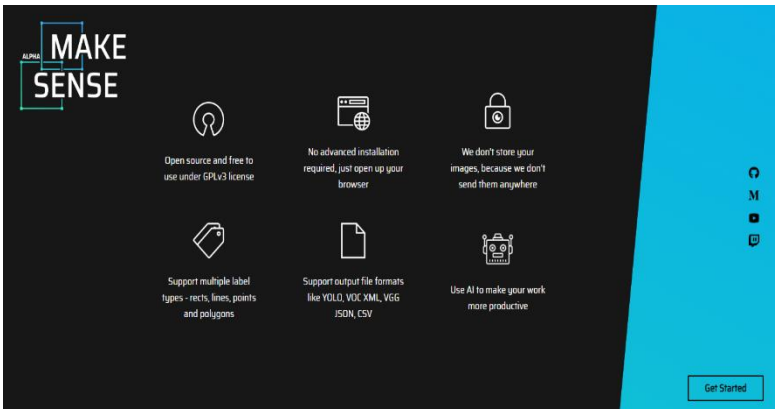
3.5. Herramienta de etiquetado y base de datos

3.5.1. MakeSense

Es una herramienta online de uso gratuito para etiquetar imágenes mediante recuadros dimensionados por el usuario. No requiere de ninguna instalación de software, tampoco importa que sistema operativo este utilizando. Es perfecto para realizar proyectos de aprendizaje profundo de visión por computadora. La figura 16 muestra la interfaz de desarrollo de la herramienta.

Figura 16

Herramienta MakeSense



Nota. Obtenida de MakeSense.ia

3.5.1.1. Cuadros limitadores

La herramienta makeSense utiliza cuadros limitadores (Bounding Boxes) en donde se proporciona una percepción visual del objeto. Además, se define áreas precisas que el modelo debe aprender a reconocer. En el contexto de Yolo, estos cuadros limitadores se utilizan de manera audaz, En lugar de realizar operaciones complejas directas con los cuadros, utiliza una red neuronal convolucional para procesar los cuadros y la imagen de manera simultánea. En la figura 17 muestra como Yolo crea cuadros en toda la imagen con el fin de realizar predicciones sobre la presencia del objeto en esas regiones.

Figura 17

Cuadros limitadores de Yolo

Los cuadros de color naranja y rojo son las etiquetas que se han realizado en la imagen, mientras que los cuadros de color negros son los cuadros que Yolo realiza para la predicción. La red neuronal convolucional predice cuadros delimitadores junto con las probabilidades de que esos cuadros contengan el objeto a clasificar. Los cuadros que no contienen el objeto o con probabilidades muy bajas, no se eliminan o modifican directamente, sino que se utilizan para

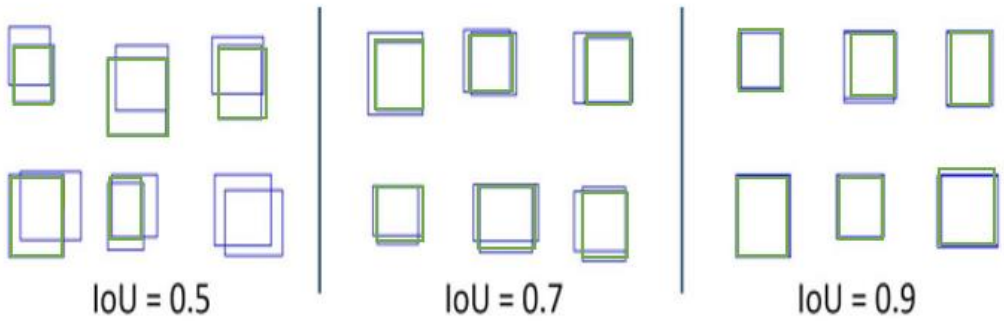
hacer predicciones. Yolo realiza un proceso de supresión de no máximos (NMS) después de realizar las predicciones, con esto, se suprimen cuadros redundantes dejando los usuarios más confiables para cada objeto.

3.5.1.2. Intersección sobre unión (IoU)

Durante el entrenamiento, Yolo aplica una medida de la superposición entre dos cuadros delimitadores, La figura 18 muestra como dos delimitadores se superponen entre sí.

Figura 18

Intersección sobre unión



Nota. Obtenida de MakeSense.ia

Este valor va de 0 a 1, donde:

- IoU > 0.5: decente
- IoU > 0.7: muy bueno
- IoU > 0.9: casi perfecto

El modelo se entrena para predecir cuadros delimitadores basándose en las etiquetas proporcionadas por la base de datos. En la figura 18, se puede apreciar que dos cuadros que se superpone, el cuadro de color rojo es el cuadro proporcionado por la herramienta MakeSense mientras que el de color azul es una predicción dada por Yolo. Entre más cercano a 1 sea el

valor de IoU, más probabilidad hay de que el modelo detecte correctamente el objeto disminuyendo falsos positivos.

3.5.1.3. Supresión de no máximos (NMS)

Este algoritmo compara IoU entre toda la detección ordenadas según su probabilidad de confianza. Posteriormente, se compara los IoU con las detecciones y si el IoU es mayor que un umbral predefinido, se elimina los cuadros delimitadores con menor confianza. Esto se aplica para eliminar cuadros redundantes durante el entrenamiento y la inferencia del modelo en tiempo real.

3.5.2. COCO dataset

Coco es un conjunto de datos de referencia para la detección de objetos que proporciona acceso a las siguientes funciones:

- Segmentación de objetos
- Mas de 300000 imágenes (220000 están categorizadas)
- 80 categorías de objetos
- 5 posibilidades de títulos de imagen

Se utiliza esta base de datos para colocar las clases del proyecto.

3.6. Entrenamiento del Modelo

3.6.1. Selección del modelo

Se selecciono los modelos Yolo y Tachable Machine para este proyecto por razones fundamentales. En primer lugar, Yolo destaca por su capacidad única para realizar detecciones rápida y precisa de múltiples objetos en una sola pasada, esto hace que sea especialmente eficiente en tiempo real. Además, se optó por Teachable Machine debido a su enfoque accesible

y educativo para entrenar modelos de aprendizaje automático. Esta plataforma simplifica el proceso de entrenamiento.

3.6.2. Creación de la base de datos

La creación de una base de datos es esencial para el desarrollo del modelo de detección de señales de tránsito de Ecuador. Se ha implementado un enfoque estratégico, dividiendo la captura de imágenes en tres zonas claves:

- Zona 1: lugares como escuelas y calles secundarias, seleccionados por la presencia de señales preventivas y reglamentarias. Se tomo alrededor de 450 fotos.
- Zona 2: se centra en áreas urbanas como el centro, mercado y malecón de la libertad, donde se encuentran señales preventivas y limitaciones de velocidad. Aproximadamente se tomaron unas 300 fotos.
- Zona 3: correspondiente a vías principales, donde se obtuvieron alrededor de 250 fotos.

Para enriquecer la base de datos y garantizar la adaptabilidad del modelo, se incorporó fotos de señales de tránsito obtenidas de la web, además, se realizaron capturas en un entorno de pruebas.

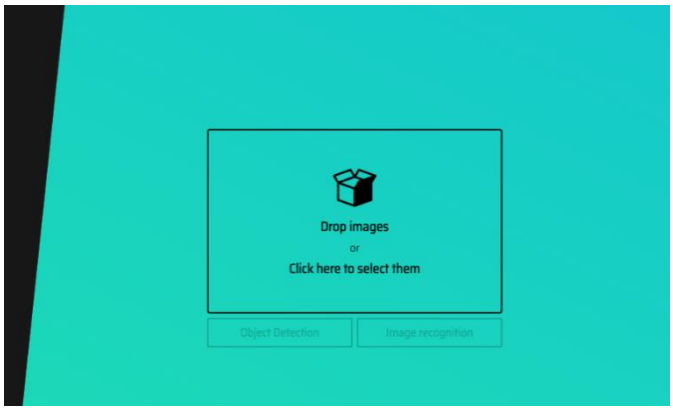
3.6.3. Etiquetado de la base de datos

El proceso de etiquetado se lo lleva mediante 4 pasos:

- Subir el conjunto de datos: se inicia sesión en MakeSense.ia y se crea un nuevo documento en donde se sube el conjunto de imágenes, la interfaz de la herramienta se muestra en la figura 19.

Figura 19

Interfaz de MakeSense

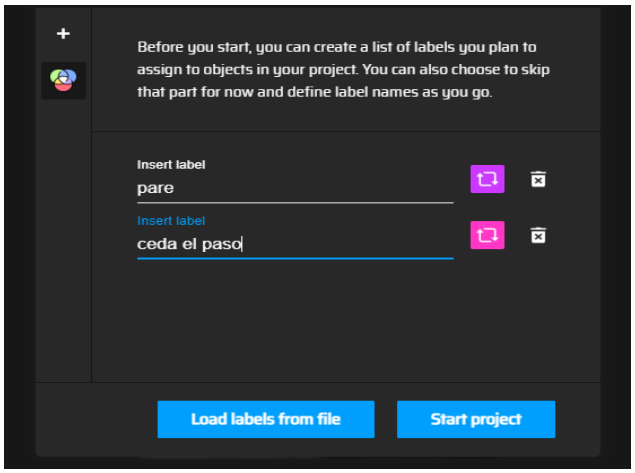


Nota. Obtenida de MakeSense.ia

- Agregar etiquetas: en la figura 20, se observa las etiquetas agregadas de acuerdo a los objetos que desea detectar

Figura 20

Agregación de etiquetas

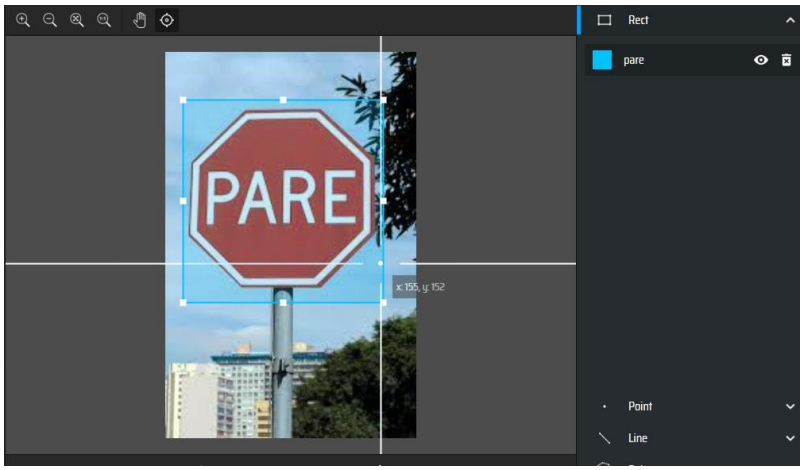


Nota. Obtenida de MakeSense.ia

- Etiquetar objetos: Se selecciona una imagen y se utiliza las herramientas de etiquetado para marcar los objetos de interés como indica la figura 21.

Figura 21

Etiquetado de objetos

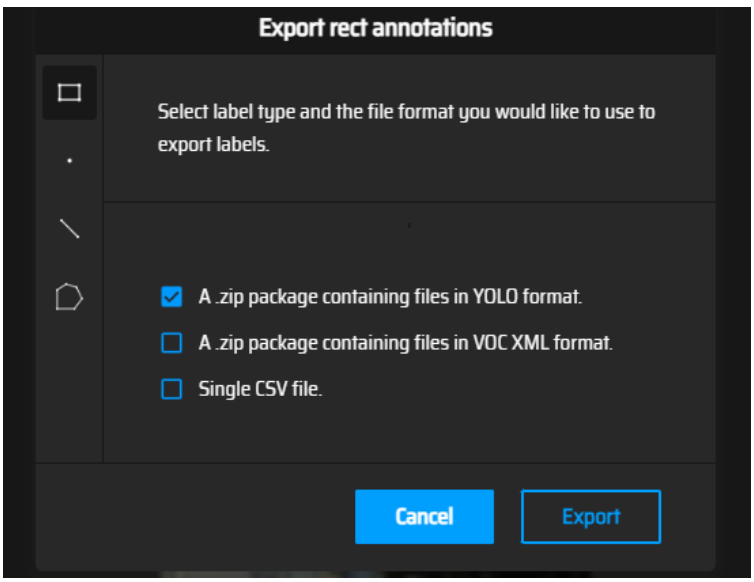


Nota. Obtenida de MakeSense.ia

- Descargar etiquetas: se busca la opción exportar y se descarga las etiquetas en formato para Yolo (como txt) como se observa en la figura 22.

Figura 22

Descarga de conjunto de etiquetas



Nota. Obtenida de MakeSense.ia

Cada imagen debe tener su etiqueta correspondiente, por ende, cada una debe tener un nombre o numeración específica para que no haya problemas al momento de entrenar el modelo.

3.6.4. Entrenamiento con Yolov5n

Para el entrenamiento, se emplea el entorno de desarrollo de Google Colab, utilizando un cuaderno accesible a través del siguiente enlace: https://colab.research.google.com/drive/1kA467UqwxXjDdWVBxvNF0R3bNx0FnRZl?usp=drive_link. En el entorno de programación seleccionado, se encuentran los requisitos necesarios para YOLOv5, así como la base de datos creada para el proceso de entrenamiento.

3.6.4.1. Parámetros de entrenamiento

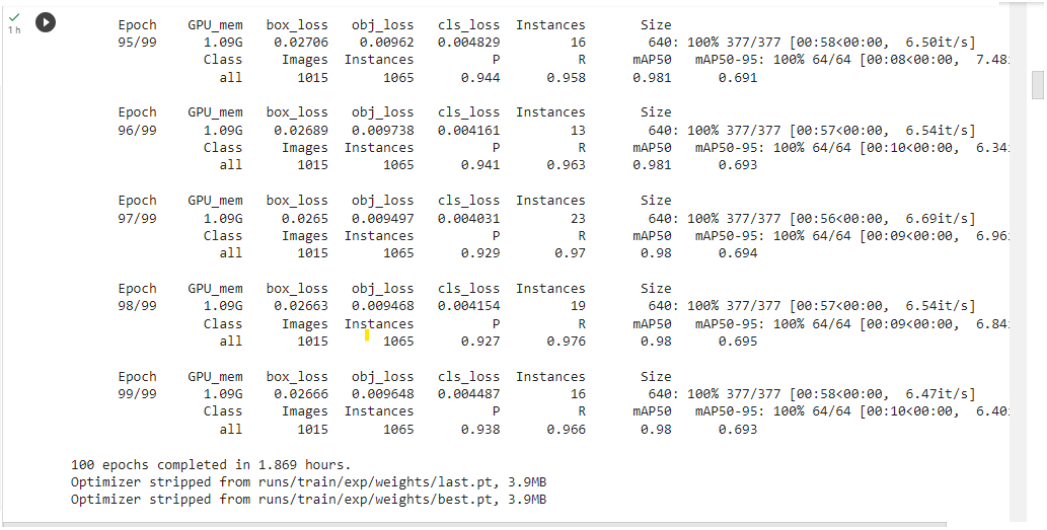
Los parámetros de entrenamiento han sido seleccionados de acuerdo con la investigación realizada, optando por los siguientes:

- **Épocas**: Se estableció un valor de 100 épocas para el entrenamiento del modelo, considerando que se trata de un modelo de rendimiento medio.
- **Batch:** Se fijo un valor de 32, indicando que en cada paso de entrenamiento se utilizarán 8 imágenes, ajustándose a las condiciones y especificaciones de la capacidad computacional disponible.
- **Tamaño de imagen:** Este parámetro se mantiene en su valor predeterminado de 640 pixeles en su lado más largo, conforme a la configuración estándar de cada modelo de Yolo.

Tras realizar los procesos descritos anteriormente y los propuestos en el Google Colab. Se inicia el proceso de entrenamiento, en el cual se calcularán las etapas necesarias para la convergencia del modelo, teniendo en cuenta el uso del espacio en el disco duro y la memoria empleada. En la figura 23 se observa el proceso de entrenamiento.

Figura 23

Entrenamiento de Yolov5n

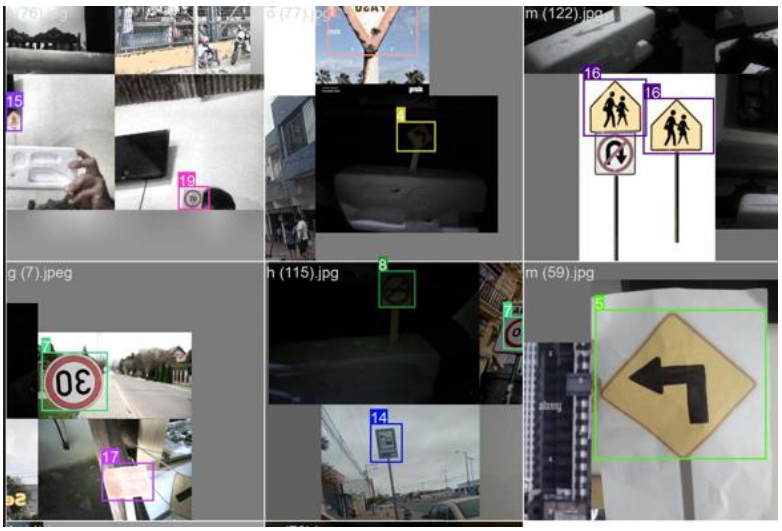


```
     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     95/99      1.09G    0.02706    0.00962   0.004829         16        640: 100% 377/377 [00:58<00:00,  6.50it/s]
                Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 64/64 [00:08<00:00,  7.48
                  all       1015       1065      0.944      0.958      0.981      0.691

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     96/99      1.09G    0.02689   0.009738   0.004161         13        640: 100% 377/377 [00:57<00:00,  6.54it/s]
                Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 64/64 [00:10<00:00,  6.34
                  all       1015       1065      0.941      0.963      0.981      0.693

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     97/99      1.09G     0.0265   0.009497   0.004031         23        640: 100% 377/377 [00:56<00:00,  6.69it/s]
                Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 64/64 [00:09<00:00,  6.96
                  all       1015       1065      0.929       0.97       0.98      0.694

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     98/99      1.09G    0.02663   0.009468   0.004154         19        640: 100% 377/377 [00:57<00:00,  6.54it/s]
                Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 64/64 [00:09<00:00,  6.84
                  all       1015       1065      0.927      0.976       0.98      0.695

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     99/99      1.09G    0.02666   0.009648   0.004487         16        640: 100% 377/377 [00:58<00:00,  6.47it/s]
                Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 64/64 [00:10<00:00,  6.40
                  all       1015       1065      0.938      0.966       0.98      0.693

100 epochs completed in 1.869 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 3.9MB
Optimizer stripped from runs/train/exp/weights/best.pt, 3.9MB
```

El proceso de entrenamiento se puede visualizar en la figura 24, Yolo identifica todos los objetos presentes en la imagen de la base de datos proporcionada mediante coordenadas de los cuadros delimitadores (bounding boxes). Esto ayuda al modelo a comprender la ubicación espacial de los objetos en cada imagen.

Figura 24

Validación de etiquetas de Yolov5n



Terminado el proceso entrenamiento, Yolo realiza validaciones en donde en cada imagen realiza una predicción como se observa en la figura 25. Yolo enumera las clases a la cual pertenece el objeto, así mismo, muestra la probabilidad de que la detección sea correcta la cual esta expresada con un valor entre 0 a 1, donde 1 indica una alta confianza en la predicción.

Figura 25

Predicciones de Yolov5n

3.6.5. Entrenamiento con Yolov8x

Para el entrenamiento, se emplea el entorno de desarrollo de Google Colab, utilizando un cuaderno accesible a través del siguiente enlace: https://colab.research.google.com/drive/1ELHVcWcL6RDF1TdhJ2_Xyddxsuebr33b?usp=drive_link. En el entorno de programación seleccionado, se encuentran los requisitos necesarios para YOLOv8, así como la base de datos creada para el proceso de entrenamiento.

3.6.5.1. Parámetros de entrenamiento

A diferencia de yolov5 donde configuramos tres parámetros, yolov8 requiere dos configuraciones:

- **Épocas**: se estable un valor de 20 épocas para el entrenamiento ya que es un modelo de alto rendimiento, pero con cargas computacionales altas.
- **Tamaño de imagen**: este parámetro se deja por defecto en 640 pixeles en su lado más largo que es valor dado en cada modelo de Yolo.

Tras realizar los procesos descritos anteriormente y los propuestos en el Google Colab. Se inicia el proceso de entrenamiento, en el cual se calcularán las etapas necesarias para la convergencia del modelo, teniendo en cuenta el uso del espacio en el disco duro y la memoria empleada.

El proceso de entrenamiento de Yolov8 es idéntico al de Yolov5. En la figura 26 se observa el entrenamiento donde se identifica los objetos de la imagen utilizando las coordenadas de los bounding boxes que rodena cada objeto. Posteriormente, en la figura 27 se visualiza las predicciones realizadas y las probabilidades que cada predicción sea correcta.

Figura 26

Validación de etiquetas de Yolov8x

Figura 27

Predicciones de Yolov8x

3.6.6. Entrenamiento con Teachable Machine

Para el entrenamiento, se utiliza la página de Teachable Machine del siguiente enlace: https://teachablemachine.withgoogle.com/. Teachable Machine es un modelo que proporciona las siguientes configuraciones:

- **Épocas:** se escoge un valor de 100 épocas debido a que es un modelo de rendimiento medio.
- **Tamaño de lote:** se establece un valor de 16 ya que al utilizar un tamaño de batch mayor puede provocar la paralelización de operaciones en hardware.
- **Tasa de aprendizaje**: Una tasa de aprendizaje más alta, como 0.001, puede permitir que el modelo converja más rápido durante el entrenamiento.

En la figura 28 se puede observar las detecciones que el modelo realiza, Teachable Machine nos da la opción de descargar los pesos del entrenamiento con la cual podemos realizar pruebas.

Figura 28

Predicción de Teachable Machine

3.7. Selección de sistema embebido

La elección de un sistema embebido depende de varios factores como: requisitos del proyecto, potencias de procesamiento, tamaño y herramientas de desarrollo. Dicho esto, el sistema embebido que se ha escogido para la implementación del modelo es un Raspberry pi 4 modelo B. Aquí se presenta algunas razones para la elección de este sistema:

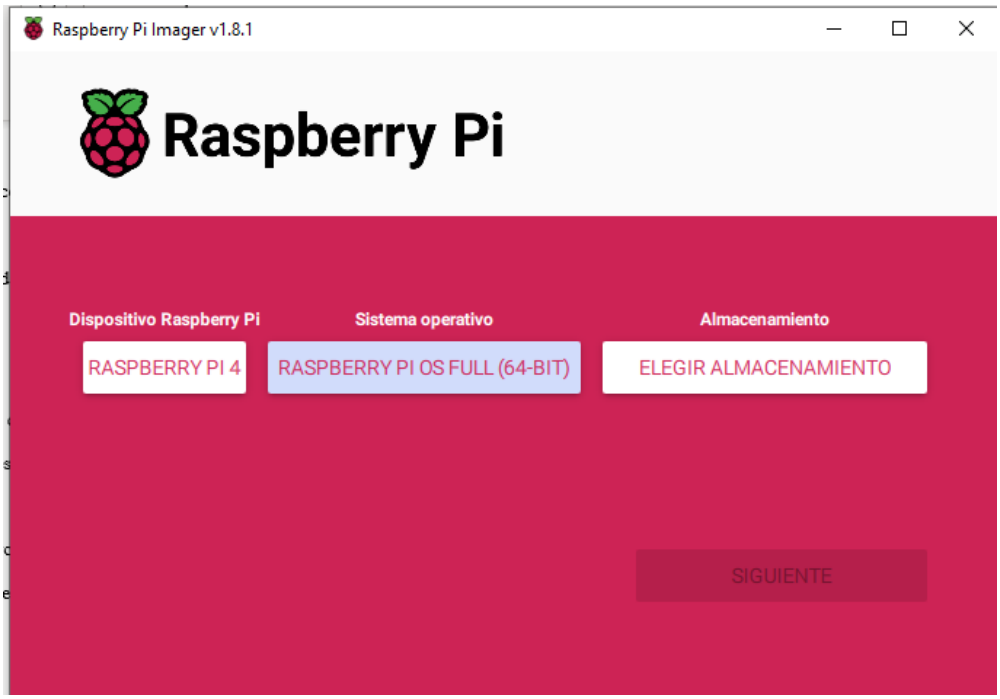
- Potencia de procesamiento: tiene un procesador quad-core AEM Cortex A72 a 1.5 Ghz con una memoria RAM de 4 Gb. Estas características son las adecuadas para implementar un modelo yolov5.
- GPIO (entrada/salida): permite interacción con dispositivos y periféricos personalizados
- Gráficos y resolución: capacidad de admitir dos monitores 4k través de salidas HDMI.

3.7.1. Instalación de software

Para la instalación del sistema operativo de raspberry, se debe dirigir a la página oficial de raspberry y descargar Raspberry Pi Imager de acuerdo al sistema operativo que se esté utilizando. Posteriormente, debe elegir el dispositivo, sistema operativo Pi OS y el almacenamiento como se observa en la figura 29. Rasberry pi tiene diferentes sistemas operativos, pero como recomendación se instale Raspberry Pi OS Full (64 bit). Sin embargo, esto dependerá de la tarjeta de almacenamiento.

Figura 29

Raspberry Pi OS Imager



Se debe tener en cuenta que las tarjetas SD son un componente esencial en las Raspberry Pi, ya que se utilizan para almacenar el sistema operativo, programas, etc. A continuación, se presenta algunas características que se debe considerar para la elección de la misma.

- Capacidad: se recomienda una tarjeta de al menos 8 Gb para un sistema operativo con el cual se vaya a realizar tareas básicas, dependiendo de la tarea a realizar el

almacenamiento de la tarjeta debe ser más amplio. Para el proyecto se recomienda una tarjeta de al menos 32 Gb.

- Velocidad de lectura: las tarjetas se clasifican por sus velocidades de lectura, para la implementación se necesita de una SD de 100 MB/S.
- Clase: se refiere a la velocidad de escritura. Una tarjeta de clase 10 tiene 10 MB/s lo que es aceptable para el proyecto.

CAPÍTULO IV

4. Resultados del entrenamiento

4.1. Resultados de entrenamiento de modelos basados en YOLO

En la tabla 5 y 6 se representan los resultados de validación de los modelos yolov5n y yolov8x, donde:

- **Clases:** nombre de las clases que el modelo detecta.
- **Imágenes:** número de imagen que modelo toma para validación.
- **Instancias:** número de veces que el objeto ha sido detectado en las imágenes.
- **Precisión (P):** porcentaje de predicciones positivas que el modelo tiene en dicha clase, este porcentaje va de 0 a 1.
- **Recall (R):** porcentaje de sensibilidad que tiene el moldeo en dicha clase.
- **mAP_50:** precisión promedio considerando la superposición con un umbral de intersección del 50 %.
- **mAP50-90:** precisión promedio considerando la supervisión con un umbral de intersección del 50 % a 90%.

Tabla 5

Resultados de Yolov5n

Clases	imágenes	Instancias	P	R	mAP 50	mAP 50-95
Todas	**1015**	**1065**	**0.927**	**0.976**	**0.98**	**0.695**
Pare	1015	54	0.954	0.963	0.991	0.764
Ceda el paso	1015	54	0.94	1	0.98	0.73
Curva abierta a la derecha	1015	52	0.861	0.846	0.948	0.712

Curva abierta a la izquierda	1015	52	0.842	0.942	0.96	0.718
Curva cerrada a la derecha	1015	52	0.826	0.923	0.95	0.666
Curva cerrada a la izquierda	1015	52	0.942	0.931	0.965	0.708
Doble vía	1015	58	0.95	0.98	0.992	0.653
Límite de velocidad de 30km/h	1015	51	0.96	1	0.995	0.753
No adelantar	1015	59	0.963	1	0.995	0.737
No estacionar	1015	51	0.918	1	0.987	0.686
No estacionar ni adelantarse	1015	58	0.95	0.983	0.97	0.661
No girar a la derecha	1015	50	0.935	0.96	0.984	0.662
No girar a la izquierda	1015	51	0.841	1	0.951	0.635
No girar en U	1015	52	0.958	1	0.994	0.707
Parada de bus	1015	52	0.91	1	0.995	0.692
Peatones en la vía	1015	51	0.92	1	0.983	0.739
Zona escolar	1015	59	0.975	0.983	0.991	0.645
Reduzca la velocidad	1015	51	0.995	1	0.995	0.75
Una vía	1015	55	0.915	1	0.991	0.604

Límite de velocidad de 70km/h	1015	51	0.991	1	0.995	0.681

Tabla 6

Resultados de Yolov8x

Clases	imágenes	Instancias	P	R	mAP 50	mAP 50-95
Todas	**1015**	**1065**	**0.827**	**0.933**	**0.916**	**0.644**
Pare	1015	54	0.86	0.963	0.98	0.767
Ceda el paso	1015	54	0.901	0.944	0.957	0.71
Curva abierta a la derecha	1015	52	0.536	0.673	0.704	0.544
Curva abierta a la izquierda	1015	52	0.496	0.885	0.601	0.434
Curva cerrada a la derecha	1015	52	0.865	0.827	0.93	0.64
Curva cerrada a la izquierda	1015	52	0.685	0.942	0.874	0.629
Doble vía	1015	58	0.926	0.948	0.953	0.624
Límite de velocidad de 30km/h	1015	51	0.977	1	0.995	0.754
No adelantar	1015	59	0.895	0.932	0.971	0.703
No estacionar	1015	51	0.824	1	0.961	0.676
No estacionar ni adelantarse	1015	58	0.971	1	0.994	0.727

No girar a la derecha	1015	50	0.57	0.92	0.803	0.582
No girar a la izquierda	1015	51	0.545	0.804	0.679	0.432
No girar en U	1015	52	0.844	1	0.993	0.707
Parada de bus	1015	52	0.888	0.942	0.977	0.637
Peatones en la vía	1015	51	0.944	0.998	0.991	0.69
Zona escolar	1015	59	0.967	0.982	0.992	0.647
Reduzca la velocidad	1015	51	0.961	0.98	0.991	0.742
Una vía	1015	55	0.894	0.945	0.975	0.549
Límite de velocidad de 70km/h	1015	51	0.989	0.98	0.993	0.681

Como se observa en las tablas, cada clase tiene valores diferentes, esto se debe a la base de datos y las imágenes de validación. A continuación, se realiza una comparación con cuatro clases con mejores y peores resultados, tomando a consideración los valores de mAP50 y mAP50-95 como datos relevantes ya que muestran el rendimiento de cada detección en las diferentes clases del modelo.

4.1.1. Análisis de mejores resultados

La tabla 7 corresponde al modelo yolov5n en donde se presenta 4 clases con los mejores resultados de entrenamiento.

Tabla 7

Mejores resultados de Yolov5n

Clases	imágenes	Instancias	P	R	mAP 50	mAP 50-95
Límite de velocidad de 30km/h	1015	51	0.96	1	0.995	0.753
Peatones en la vía	1015	51	0.92	1	0.983	0.739
Ceda el paso	1015	54	0.94	1	0.98	0.73
No girar en U	1015	52	0.958	1	0.994	0.707

En la tabla 7 se observa que para cada clase se toma 1015 imágenes de validación, estas imágenes son proporcionadas por la base de datos creada para el entrenamiento. Además, se observa que el recall (R) tiene un valor de 1 para todas las clases, esto significa que el modelo está identificando todos los objetos positivos en el conjunto de datos, por ende, no ha tenido falsos negativos y todas las instancias de la clase de interés han sido identificadas correctamente. Sin embargo, es importante tener en consideración que un alto valor de recall viene a expensas de la precisión. Es decir, puede a ver más falsos positivos (objetos detectados incorrectamente) cuando se prioriza un alto recall.

Por otro lado, se tiene parámetros con valores variantes como las instancias, precisión (P), mAP 50 y 50-95. Las instancias son el número de veces que el objeto ha sido detectado, en este caso las instancia están entre los valores de 51 y 54, esto se debe a que la base de datos tiene un rango de 50 y 52 imágenes de validación para cada clase (señal de tránsito). En el caso de la clase “ceda el paso” tenemos 54 instancias detectadas, eso quiere decir que esta clase tiene 1 o 2 falsos positivos lo cual es aceptable tomando a consideración los demás parámetros

como la precisión que tiene un valor 0.94 y el recall un valor de 1, esto indica que, de cada 10 detecciones de la clase, 9 son verdaderos positivos y 1 puede ser un falso positivo.

Los valores de mAP50 y mAP50-95 son muy relevantes en los modelos de detección ya que nos brindan la información del rendimiento del modelo. En la tabla 7 observamos que el parámetro mAP50-95 es menor que mAP50 en todas las clases, en el caso de la clase “No girar en U” tiene un valor de mAP50-95 de 0.707, esto indica que el modelo tiene dificultades en mantener altas precisiones en umbrales del 50% al 95% de confianza provocado por las imágenes de la base de datos. La base de datos cuenta con 150 imágenes para esta clase, el valor de 0.707 se debe que las imágenes tienen objetos parcialmente visibles, pequeños y con características desafiantes, a su vez, este objeto tiene un parecido a otros objetos como indica la figura 30.

Figura 30

Similitudes de clases en Yolov5n

La herramienta MakeSense encierra el objeto en un cuadro delimitador, lo que provoca que el modelo confunda los objetos en un pequeño porcentaje. No obstante, el tener un valor mAP50-95 bajo, no es necesariamente un indicativo que el modelo sea pésimo, sino que son indicadores de ajuste y de mayor evaluación. Por otro lado, el mAP50 tiene valores más alto lo que indica que el modelo prioriza las detecciones con confianza alta en todas las clases de la tabla 7.

La tabla 8 corresponde al modelo yolov8x en donde se presenta 4 clases con los mejores resultados de entrenamiento.

Tabla 8

Mejores resultados de Yolov8x

Clases	imágenes	Instancias	P	R	mAP 50	mAP 50-95
Pare	1015	54	0.86	0.963	0.98	0.767
Límite de velocidad de 30km/h	1015	51	0.977	1	0.995	0.754
Reduzca la velocidad	1015	51	0.961	0.98	0.991	0.742
No estacionar ni detenerse	1015	58	0.971	1	0.994	0.727

Las épocas que se utilizaron en Yolov8x fueron 20 a diferencia de yolov5n que fueron 100, esto es debido a las cargas computacionales que el modelo requería. Yolov8 es un modelo de detección de alto rendimiento, lo que conlleva que los resultados sean parecidos a pesar de la diferencia de las épocas de entrenamiento.

El número de imágenes de validación es igual ya que es la misma base de datos. En el caso de la clase “no estacionar ni detenerse”, se observa que tiene 58 instancias en un conjunto de 50 a 52 imágenes de validación que contiene la base de datos, esto indica que hay 4 o 6 falsos positivos. Este valor se puede considerar levemente alto, no obstante, se observa una precisión de 0.971 con un recall (R) de 1. La precisión indica el nivel de confianza que el modelo tiene en esta clase, a su vez, el recall de 1 indica que todas las instancias han sido detectadas correctamente. En otras palabras, estos valorares indican que de cada 10 detecciones

2 o 3 son falsos positivos lo cual es aceptable. Por otro lado, la clase “Pare” tiene un valor de recall de 0.963 con 54 instancias detectadas y un rendimiento de mAP50 de 0,98, el valor de recall se debe al conjunto de imágenes ya que un pequeño porcentaje se tomó en ambientes con poca iluminación y otro con objetos diferentes.

En la evaluación de las 4 clases con los mejores valores de sus parámetros de entrenamiento, se puede decir que yolov8x demostró un rendimiento adecuado a pesar de ser entrenando con 20 épocas, sin embargo, para poner en marcha el modelo en un entorno de pruebas se necesita altos requerimientos computacionales. Por otro lado, yolov5n demostró una capacidad sólida para mantener equilibrio entre precisión, recall ofreciendo un buen rendimiento del modelo.

4.1.2. Análisis de peores resultados

La tabla 9 corresponde al modelo yolov8x en donde se presenta 4 clases con los peores resultados de entrenamiento.

Tabla 9

Peores resultados de Yolov8x

Clases	imágenes	Instancias	P	R	mAP 50	mAP 50-95
No girar a la derecha	1015	50	0.57	0.92	0.803	0.582
Curva abierta a la derecha	1015	52	0.536	0.673	0.704	0.544
Curva abierta a la izquierda	1015	52	0.496	0.885	0.601	0.434
No girar a la izquierda	1015	51	0.545	0.804	0.679	0.432

Se puede observar que los valores de los parámetros de rendimiento son muy bajos al igual que sus precisiones, en el caso de la clase “No girar a la izquierda”, se tiene un valor de 0.432 en mAP50-95, esto indica que el modelo tiene un rendimiento muy bajo cuando consideramos umbrales altos de confianza, es decir, el modelo tiene dificultades para ajustarse a la variabilidad en la superposición entre las cajas de predicción y las cajas de referencia. El valor de 0.432 podría atribuirse a dos posibles factores en el conjunto de datos. Primero, la presencia de similitudes visuales entre las clases como indica la figura 31. Segundo, las variaciones de iluminación de las imágenes contribuyen a un bajo rendimiento ya que el modelo tiende a detectar falsos positivos. Por otro lado, el valor de recall para esta clase es de 0.804, lo que indica que el modelo está detectando un 80% de todas las instancias. Sin embargo, el hecho de tener una precisión de 0.545 y un valor de recall alto, significa que el modelo está detectando falsos positivos, es decir, el modelo está prediciendo objetos que en realidad no son correctos.

Figura 31

Similitudes de clases en Yolov8x

Si se aplica la misma analogía en las 4 clases del modelo yolov5n, se puede realizar una comparación de los modelos. La tabla 10 corresponde al modelo yolov5n en donde se presenta 4 clases con los peores resultados de entrenamiento.

Tabla 10

Peores resultados de Yolov5n

Clases	imágenes	Instancias	P	R	mAP 50	mAP 50-95
No girar a la derecha	1015	50	0.935	0.96	0.984	0.662
Una via	1015	55	0.915	1	0.991	0.604
Curva cerrada a la derecha	1015	52	0.826	0.923	0.95	0.666
No girar a la izquierda	1015	51	0.841	1	0.951	0.635

Los resultados de Yolov5n, a pesar de ser considerados los peores resultados, se observa que no son inferiores del 0.6 en términos de mAP50-95. En el caso de la clase “no girar a la derecha”, tiene un valor de recall de 0.96 con 50 instancias detectadas, el rango de imágenes de detección va de 50 a 52 imágenes, lo que indica que el modelo tiende a equivocarse en 1 o 2 instancias con esta clase. A pesar de ello, el modelo mantiene una precisión del 0.953 y un rendimiento de 0.984 con umbrales del 50 % de confiabilidad. Estos valores tienen una mejora drástica en comparación con los valores de Yolov8x y a pesar de tener la misma equivocación debido a sus similitudes, Yolov5n es capaz de identificar con mayor precisión las clases manteniendo una sensibilidad de reconocimiento a umbrales del 50% a 95% de confiabilidad.

4.1.3. Precisión y confianza

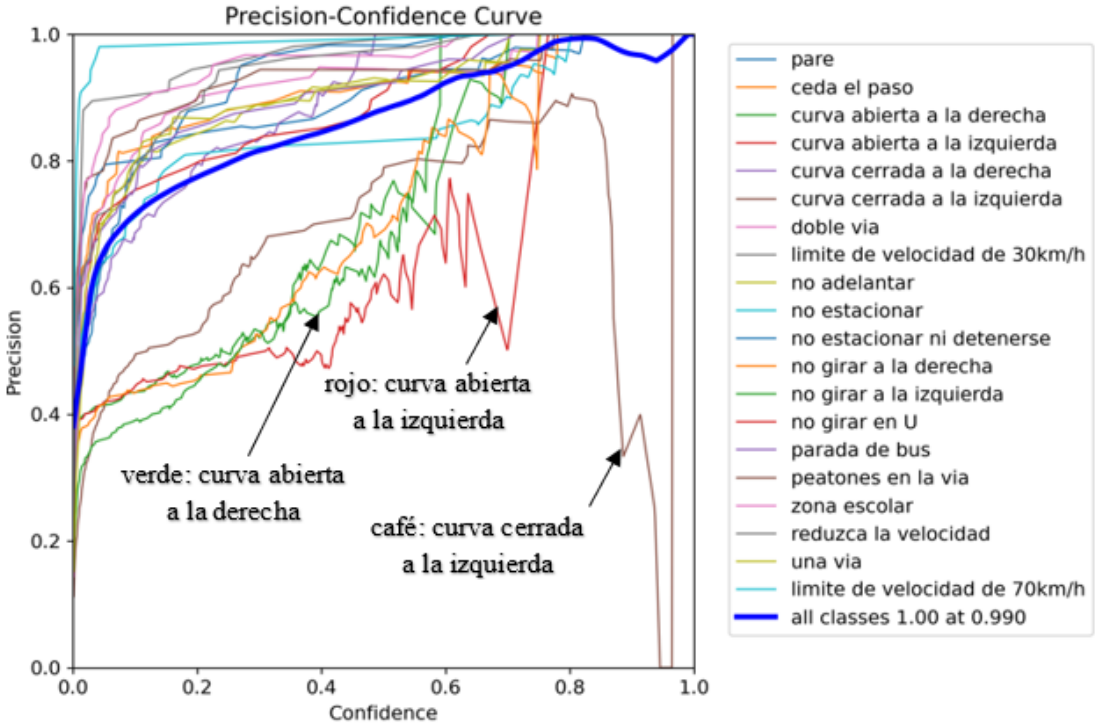
La curva “Precisión-Confianza” se traza típicamente para cada clase conforme el modelo sea entrenado, pero también es posible tener una única agregada de color azul como se observa en la figura 32. Esta línea “all class” representa el rendimiento promedio de todas las clases del modelo a medida que se ajusta el umbral de confianza, no establece un límite específico para considerar al modelo como eficiente o ineficiente. Sin embargo, se puede realizar las siguientes interpretaciones:

- **Líneas por encima de “all class”:** representan las curvas para las clases individuales que tiene un rendimiento superior al promedio.
- **Líneas por debajo de “all class”:** representan las curvas de cada clase que tiene un rendimiento inferior al promedio.

En esta curva se puede apreciar que las curvas de un cierto grupo de clases están por debajo de la línea promedio, mientras que la mayoría de clases se encuentran en un alto rendimiento.

Figura 32

Precisión y confianza de Yolov8x



Se ha señalado 3 clases en donde el rendimiento del modelo no es aceptable. En las curvas señaladas se aprecia un aumento de precisión conforme aumenta el umbral de confianza, esto se debe a que cuando se tiene un umbral de confianza bajo, el modelo es más propenso a tener predicciones positivas incluso cuando no está seguro. En otras palabras, el modelo clasifica instancias como verdaderos positivos aun teniendo una precisión baja. Sin embargo, las curvas de color rojo y café pertenecientes a la clase “curva abierta a la izquierda” y “curva cerrada a la izquierda” tiene picos de pérdida de presión a un umbral alto de confianza.

La clase “curva cerrada a la izquierda” tiene un pico de pérdida más pronunciado, esto se debe a que cuando el umbral de confianza aumenta, el modelo se vuelve más selectivo y solo clasifica instancias positivas cuando está seguro de que es verdadera. Esto puede ser de gran beneficio, ya que el modelo tiene menos falsos positivos, no obstante, esta situación provoca que clasifique instancias positivas como negativas por el hecho de ser muy difícil de predecir. Este pico de pérdida es producido por el conjunto de imágenes de entrenamiento, como se ha dicho anteriormente, las clases “curva cerrada a la derecha”, “curva cerrada a la izquierda”, “curva abierta a la derecha” y “curva abierta a la izquierda” tienen un parecido y le es un poco difícil detectar al modelo. Otra dificulta que tiene el modelo es que un cierto número de imágenes se tomó con poca iluminación como muestra la figura 33.

Figura 33

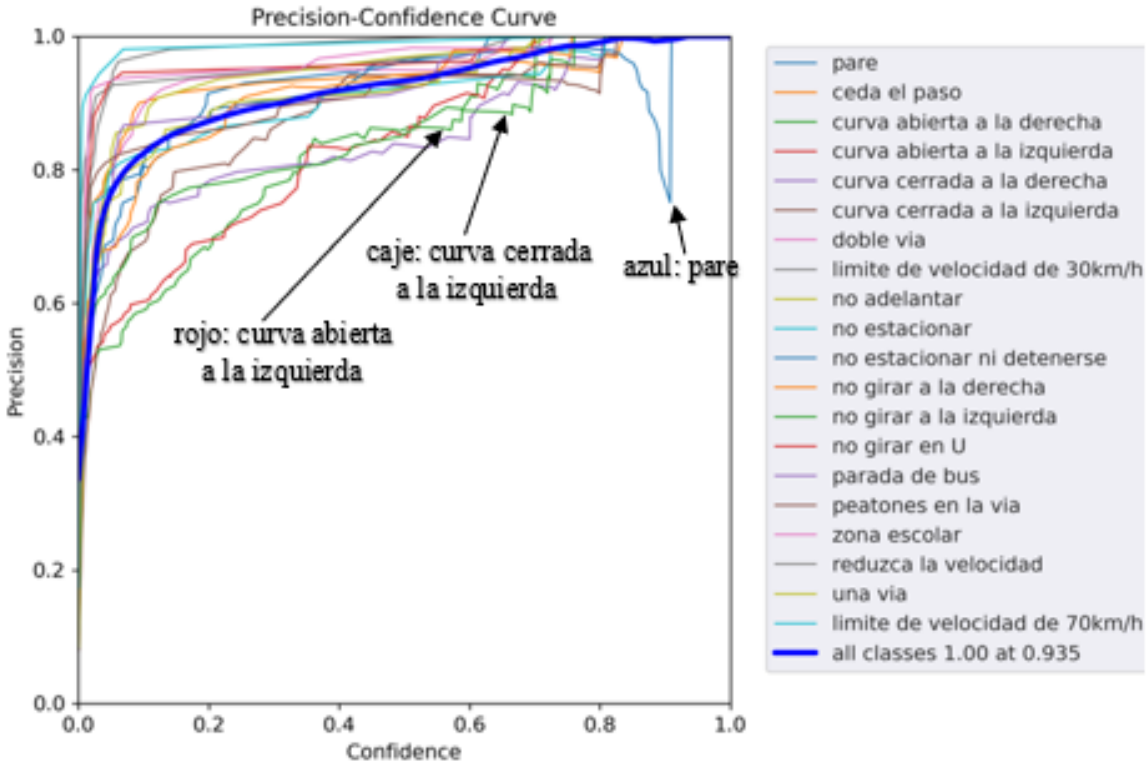
Similitudes en las clases de curvas abiertas

Otro dato particular que se observa en la figura 32 es que la curva de la clase “curva abierta a la izquierda” tiene un pico de pérdida, pero conforme aumenta el umbral va recuperando su precisión y su longitud es más corta que la curva de color café. La recuperación de precisión se debe a que el modelo se expuso a más datos de la clase, en donde aprendió patrones adicionales mejorando su capacidad para hacer predicciones precisas. Mientras que la longitud corta de la curva es provocada por las dificultades que tiene el modelo en detectar dicha clase en umbrales de confianza más altos.

Por otro lado, la figura 34 muestra la curva de “precisión-confianza” de yolov5n. en esta figura se observa una mejora radical en todas las clases, teniendo clases por debajo del rendimiento promedio, pero con valores de precisión y confianza aceptables.

Figura 34

Precisión y confianza de Yolov5n



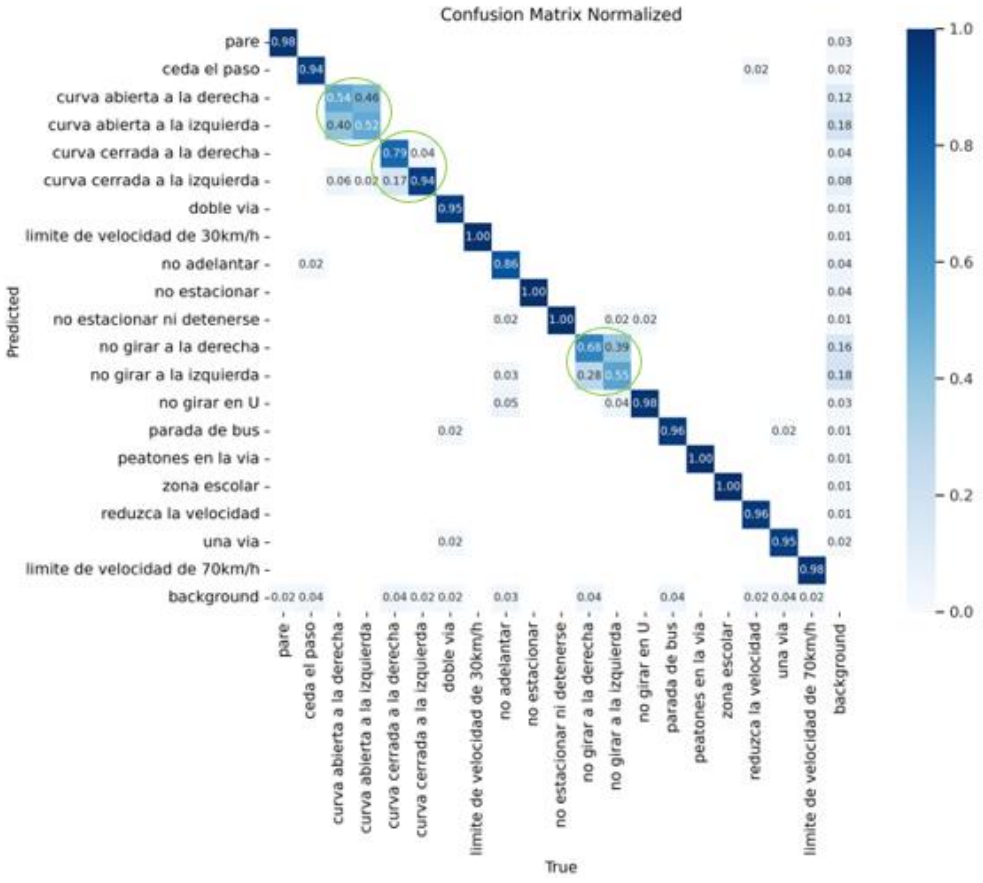
Como se puede observar, las curvas de color café y rojo con picos de pérdidas en yolov8x mejoraron drásticamente en yolov5n, esto debido que el modelo de yolov5n se entrenó con 100 épocas mejorando el modelo en detección es de patrones más complejos y sutiles en el conjunto de imágenes. Sin embargo, el entrenar un modelo con muchas épocas puede aumentar el riesgo de que los pesos se ajusten exactamente a los datos de entrenamiento, es decir, el modelo va a detectar objetos que sea exactos a los objetos de entrenamiento y provoque dificultades para detectar los objetos en imágenes, lugares o posiciones diferentes.

4.1.4. Matriz de confusión

La matriz de confusión es una herramienta clave para analizar el rendimiento del modelo. En la figura 35 se tiene la matriz de confusión del modelo yolov8x, en donde el eje "x" tiene los datos verdaderos y en el eje "y" los datos predichos, estos valores van de 0 a 1 siendo 1 el valor más alto en donde se puede decir que el modelo predice correctamente un valor verdadero sin errores.

Figura 35

Matriz de confusión de Yolov8x



En la figura 35 se marcó tres puntos de interés en donde se observa que el modelo ha tenido más confusiones, en el caso de la clase “curva abierta a la derecha” vemos que tiene un valor de predicción de 0.54. Sin embargo, el modelo confunde esta clase con la clase “curva abierta a la izquierda” con 0.40 y con la clase “curva cerrada a la izquierda” con 0.06. Estas confusiones ocurren de debido a similitud visual que tienen estas clases en su estructura, como se ha dicho anteriormente, la herramienta MakeSense dibuja cuadros limitadores alrededor del objeto y el modelo se encarga de aprender a identificar lo que hay adentro de este cuadro. La figura 36 muestra la similitud que existe en las 4 clases que muestra mayor confusión.

Figura 36

Similitud entre clases de curvas abiertas y cerradas

De la misma manera, se observa que en la clase “no girar a la derecha” existe confusión con la clase “no girar a la izquierda” con 0.28, y de la misma manera esta clase se confunde con la clase “no adelantar” con un valor de predictivo de 0.03. Estas confusiones se deben a las similitudes que existen en estas clases, como se observa en la figura 37.

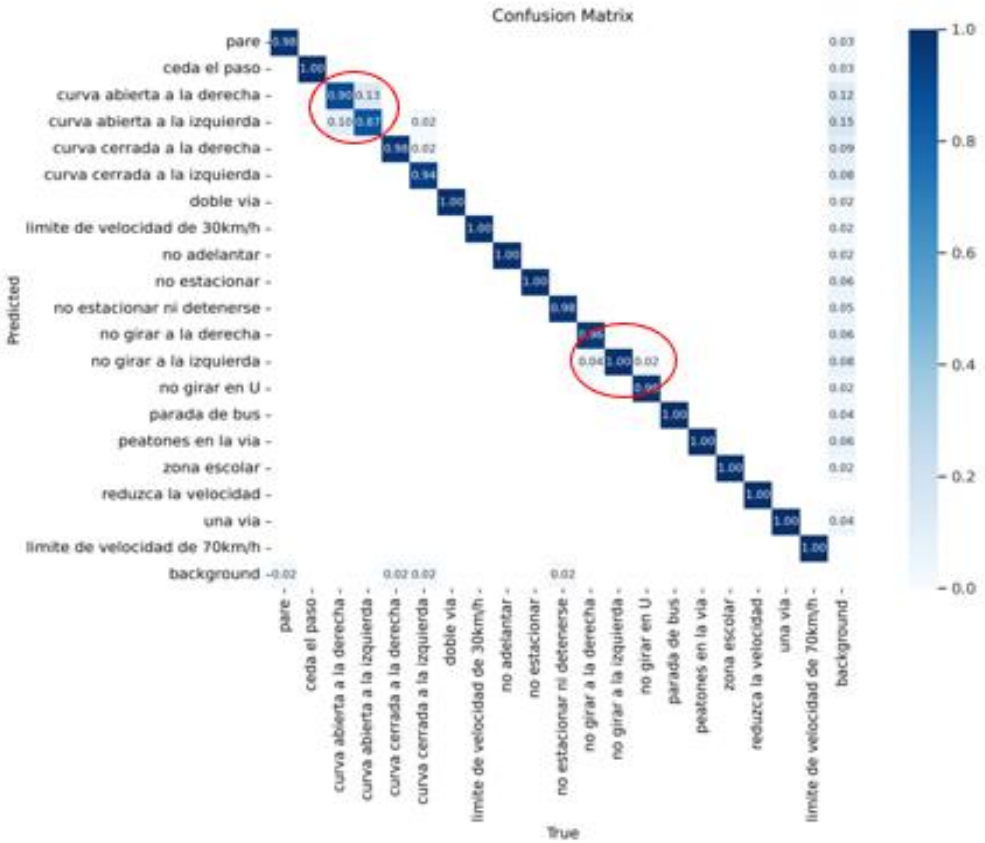
Figura 37

Similitud en clases de prohibición

Por otro lado, la figura 38 muestra la matriz de confusión del modelo yolov5n en donde se observa una mejora en los puntos donde yolov8x presenta confusiones.

Figura 38

Matriz de confusión de Yolov5n



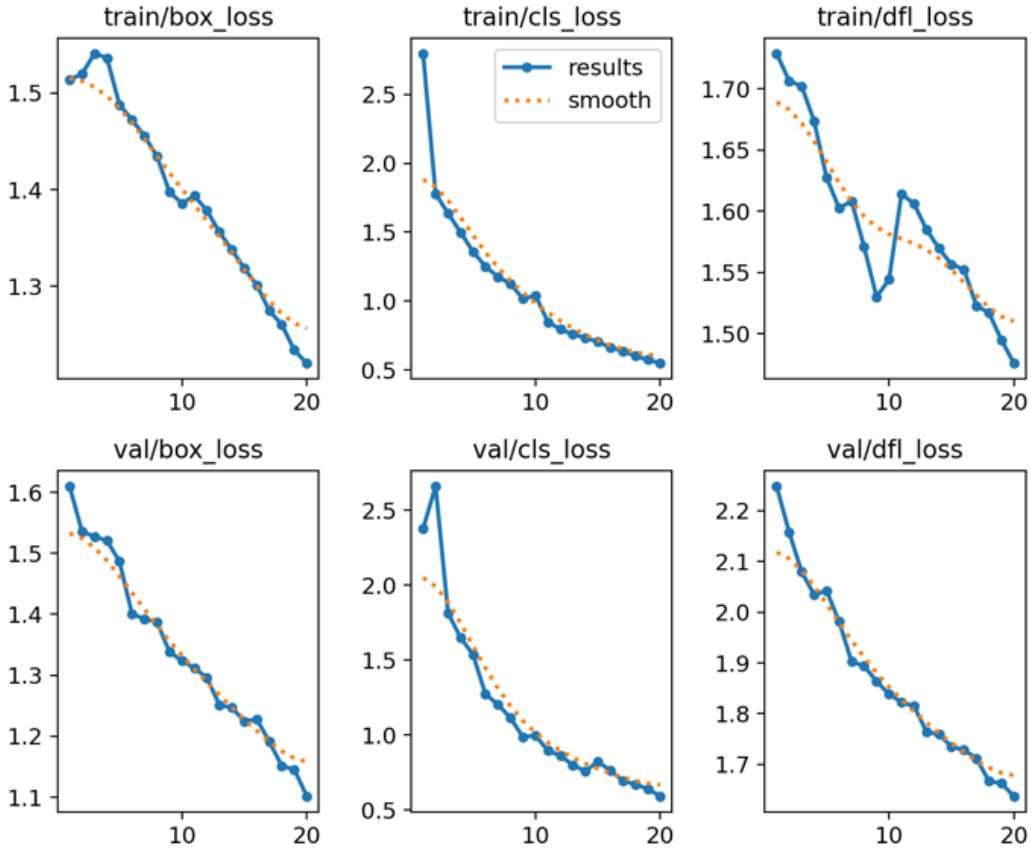
En esta matriz se observa dos puntos en donde el modelo tiene confusiones, a pesar de ello, los valores que presenta en las confusiones son bajos. En el caso de la clase “curva abierta a la derecha”, se observa que sigue teniendo una confusión con la clase “curva abierta a la izquierda”, sin embargo, el valor de esta confusión se redujo a un 0.10 a diferencia de yolov8x que tenía un valor de 0.40. Esto quiere decir que de cada 10 detecciones que realiza el modelo, 1 instancia la confunde con la clase “curva abierta a la izquierda”, esta mejora se debo a que yolov5n se entrenó con 100 épocas a diferencia de yolov8x que fueron 20, el entrenamiento con más épocas ayudo radicalmente al modelo, así se disminuyó las confusiones del modelo en las clases con similitudes. Sin embargo, las confusiones siempre van estar presentes en el modelo.

4.1.5. Métricas de pérdida (box_loss, obj_loss, cls_loss)

En la figura 39 se tiene las métricas resultantes del entrenamiento y validación de yolov8x. En la gráfica se observa los resultados, la línea de color azul (result) representa la evolución de la pérdida a medida que aumenta las épocas de entrenamiento, y la línea entre cortada (smooth) es la versión suavizada de la métrica result, en esta versión el modelo trata de eliminar los picos de aumento y disminución de la pérdida, dejando una línea con resultados medios.

Figura 39

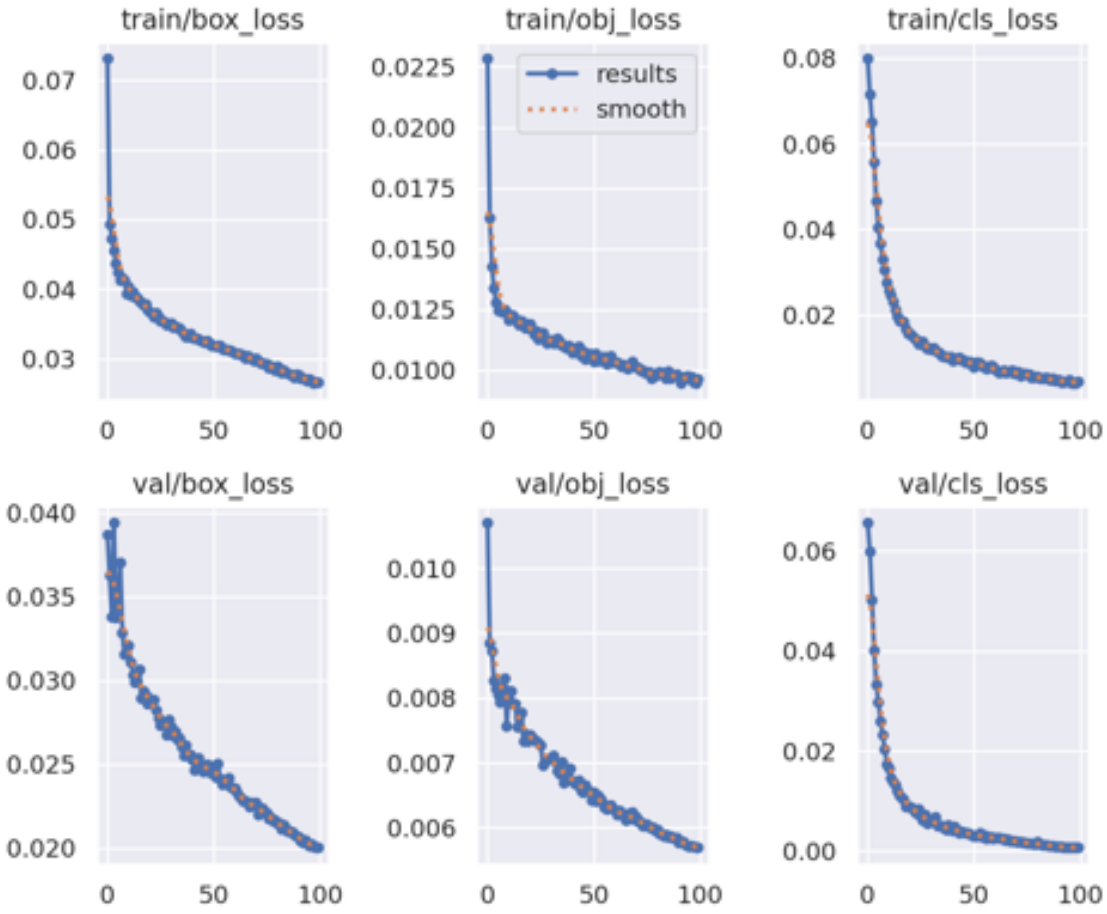
Métricas de pérdida de Yolov8x



La figura 40 muestra las métricas del modelo yolov5n, en estas métricas se puede observar una mejoría radical, las pérdidas tienen inicios menores a 0.1 y van disminuyendo conforme aumenta las épocas de entrenamiento. Otro dato que se puede observar es que la línea “smooth” es un 90 % parecida a la línea resultante, esto se debe a que las pérdidas de entrenamiento y validación son mínimas.

Figura 40

Métricas de pérdida de Yolov5n



La tabla 11 muestra los resultados de cada pérdida en su última época, Yolov8x presenta valores altos de pérdida, si bien es cierto, yolov8 es un modelo de alto rendimiento en detección de objetos, las pérdidas altas reflejadas en la tabla se deben a que el modelo fue entrenado con 20 épocas debido a los requerimientos de software del modelo. Sin embargo, sus pérdidas se acercan al valor de 1 en tan solo 20 épocas de entrenamiento, eso nos dice que si aumentamos el número de épocas el valor de sus pérdidas se va a reducir llegando al mismo o mejor nivel que yolov5n.

Yolov5n es un modelo de rendimiento medio por que necesita de más épocas de entrenamiento, además, no requiere de cargas computacionales altas.

Tabla 11

Comparación de resultados de las últimas épocas

Pérdida	Yolov5n	Yolov8x
Train/box_loss	0.0266	1.2203
Train/obj_loss	0.0096	1.4759
Train/cls_loss	0.0044	0.5496
Val/box_loss	0.0200	1.1017
Val/obj_loss	0.0056	1.6374
Val/cls_losss	0.00006	0.59296

4.2. Resultados de Teachable Machine

Debido que Teachable Machine es una plataforma online, no cuenta con una fuente de resultados específicos como Yolo. Sin embargo, se puede obtener los resultados de predicción de cada objeto como se muestra en la tabla 12.

Tabla 12

Resultados de predicción de Teachable Machine

Clase	Precisión
Ceda el paso	0.91
Curva abierta a la derecha	0.83
Curva abierta a la izquierda	0.78
Curva cerrada a la derecha	0.83
Curva cerrada a la izquierda	0.83
Doble via	0.74
Límite de 30	0.96
Límite de 70	0.83
No adelantar	0.83
No estacionar	0.74

No estacionar ni detenerse	0.70
No girar a la derecha	0,83
No girar a la izquierda	0.57
Pare	0.83
Peatones en la vía	0.91
Zona escolar	0.83
Reduzca la velocidad	0.91
Una vía	0.87

La tabla 12 muestra datos de predicción muy buenos. No obstante, se debe tener en cuenta que en herramienta online no se utilizó etiquetas por ende el modelo no es un detector de objetos más bien es un clasificador de imágenes. El clasificador no utiliza cuadros limitadores, sino que asigna una etiqueta o categoría a la imagen en su totalidad, sin proporcionar información detallada sobre la ubicación específica de los objetos dentro de la imagen.

4.2.1. Análisis de mejores resultados

Como se ha dicho anteriormente, Teachable Machine es más un clasificador de imágenes que un detector de objetos. Se ha escogido 4 clases con los mejores resultados predictivos. Con estas clases se realiza una inferencia utilizando PyCharm como entorno de programación. De esta manera se logra visualizar si el modelo es eficiente, posteriormente, se obtiene el valor de recall aplicando la ecuación 1 (Recall) descrita en los apartados anteriores

En la tabla 13 se observan los resultados de las clases escogidas para la inferencia. Para realizar la inferencia, se escoge un conjunto de imágenes a fin de analizar cuantas instancias negativas obtiene el modelo.

Tabla 13

Mejores resultados de Teachable machine

Clases	imágenes	Instancias	P	R
Límite de velocidad de 30km/h	600	126	0.96	0.862
Ceda el paso	600	122	0.91	0.842
Peatones en la vía	600	110	0.91	0.789
Una vía	600	131	0.87	0.887

En la clase “Peatones en la vía” tenemos 110 instancias, debido a que se utiliza un conjunto de 150 imágenes de inferencia para cada clase, eso quiere decir que la clase “peatones en la vía” tiene 40 instancias en donde no se ha logrado clasificar correctamente la imagen. Sin embargo, sus valores de precisión y recall son aceptables, en donde indica que de cada a 10 clasificaciones que realiza el modelo, 2 o 3 clasifica de forma incorrecta.

4.2.2. Análisis de los peores resultados

En la tabla 14 se observa las clases con peores resultados, cabe recalcar que, a pesar de tener un valor aceptable de precisión, el modelo tiende a cometer errores cuando se lo implementa con datos nuevos.

Tabla 14

Peores resultados de Teachable Machine

Clases	imágenes	Instancias	P	R
No estacionar	600	97	0.74	0.738
Doble vía	600	132	0.74	0.892
No estacionar ni detenerse	600	102	0.70	0.757
No girar a la izquierda	600	123	0.57	0.847

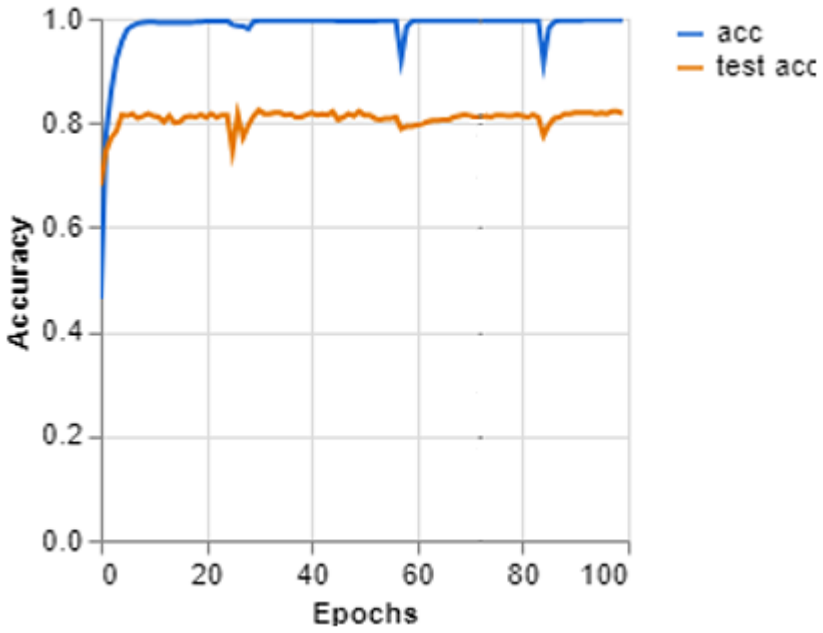
A pesar de que estas clases tienen los peores valores de entrenamiento, aun se lo puede considerar estables en ciertos puntos. En la clase “No girar a la izquierda” se tiene 123 instancias positiva y 27 instancias negativas, aparentemente el modelo no tiene muchas dificultades para clasificar esta clase, pero se observa el valor bajo de 0.57 en predicción y un valor alto de 0.847 en recall.

4.2.3. Curva de presición

Otra herramienta de evaluación que proporciona el modelo Teachable Machine, es la gráfica de precisión que se obtiene del entrenamiento. En la figura 41 se puede apreciar dos líneas, la línea de color azul (acc) representa la precisión en el conjunto de datos de entrenamiento. Mientras que la línea de color naranja (test acc) representa la precisión en un conjunto de datos de pruebas separados, es decir, imágenes de validación. Teachable Machine escoge una parte del conjunto total de imágenes que se le proporciona y las reserva para hacer validación una vez terminado el proceso de entrenamiento.

Figura 41

Curva de precisión de Teachable Machine



En la línea azul (acc) se observan dos picos de pérdida, esto se debe a cambios drásticos del conjunto de imágenes. Teachable Machine tiene varias limitaciones, una de ellas es la variabilidad de las imágenes que puede manejar, es decir, si las imágenes cambian demasiado durante el entrenamiento provoca una pérdida de precisión en el modelo. El conjunto de imágenes que se proporcionó al modelo, presenta un cambio drástico ya que se tomó imágenes con variaciones de iluminación, escenarios de prueba y algunas imágenes fueron obtenidas de red. La figura 42 muestra la variabilidad del conjunto de imágenes.

Figura 42

Variabilidad en el conjunto de imágenes

Por otro lado, la línea naranja (test acc) presenta una discusión considerable en precisión con dos picos de pérdida un poco pronunciados. La disminución de precisión en esta línea nos indica que existe un sobreajuste (overfitting) del modelo provocado por un número elevado de épocas. El modelo se adaptó demasiado a los datos de entrenamiento y tiene dificultades en clasificar nuevos datos, aun así, el modelo se encuentra en una precisión estable sobre el 0.8. La tabla 15 muestra los datos de precisión de entrenamiento y validación en la última época.

Tabla 15

Precisión en la última época

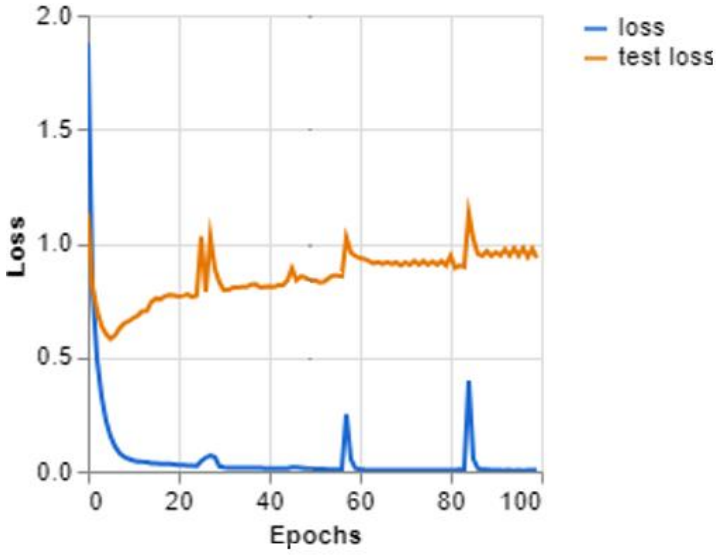
Parámetro	Época	Precisión
Precisión (acc)	100	0.997
Validación (test acc)	100	0.817

4.2.4. Pérdida de clasificación

La función de pérdida de clasificación es una medida en donde se puede visualizar que tan eficiente es el modelo al momento de clasificar el objeto en cada época del entrenamiento. En la figura 43 se observa las curvas de pérdida, la línea de color azul (loss) representa la función de pérdida en cada época de entrenamiento, mientras que la línea de color naranja (test loss) representa la pérdida de clasificación con el conjunto de imágenes de validación.

Figura 43

Pérdida de clasificación en Teachable Machine



Como se puede observar en la figura 43, la línea de pérdida de color azul (loss) presenta dos picos de aumento, anteriormente se explicó que estos picos de variación se deben a los cambios bruscos en el conjunto de imágenes de entrenamiento. El modelo se adapta tanto a un cierto tipo de imágenes que, a un cambio drástico el modelo presenta pérdida en sus clasificaciones. Al no estar seguro si la imagen pertenece a una cierta clase específica, el modelo la selecciona como una instancia negativa y a medida que el modelo aumenta sus épocas, va aumentando su nivel de confianza y logra clasificar las nuevas imágenes. Algo similar pasa en la línea naranja (test loss), se observa que la curva aumenta drásticamente su pérdida de clasificación, a su vez genera picos en donde la pérdida de clasificación del modelo sobrepasa el valor de 1. Esto quiere decir que el modelo se sobre ajustó demasiado a los datos de entrenamiento por lo que hacer expuesto a nuevos datos, no logra clasificarlos correctamente. En la tabla 16 se observa los valores de pérdida en la última época de entrenamiento.

Tabla 16

Pérdida de clasificación en la última época

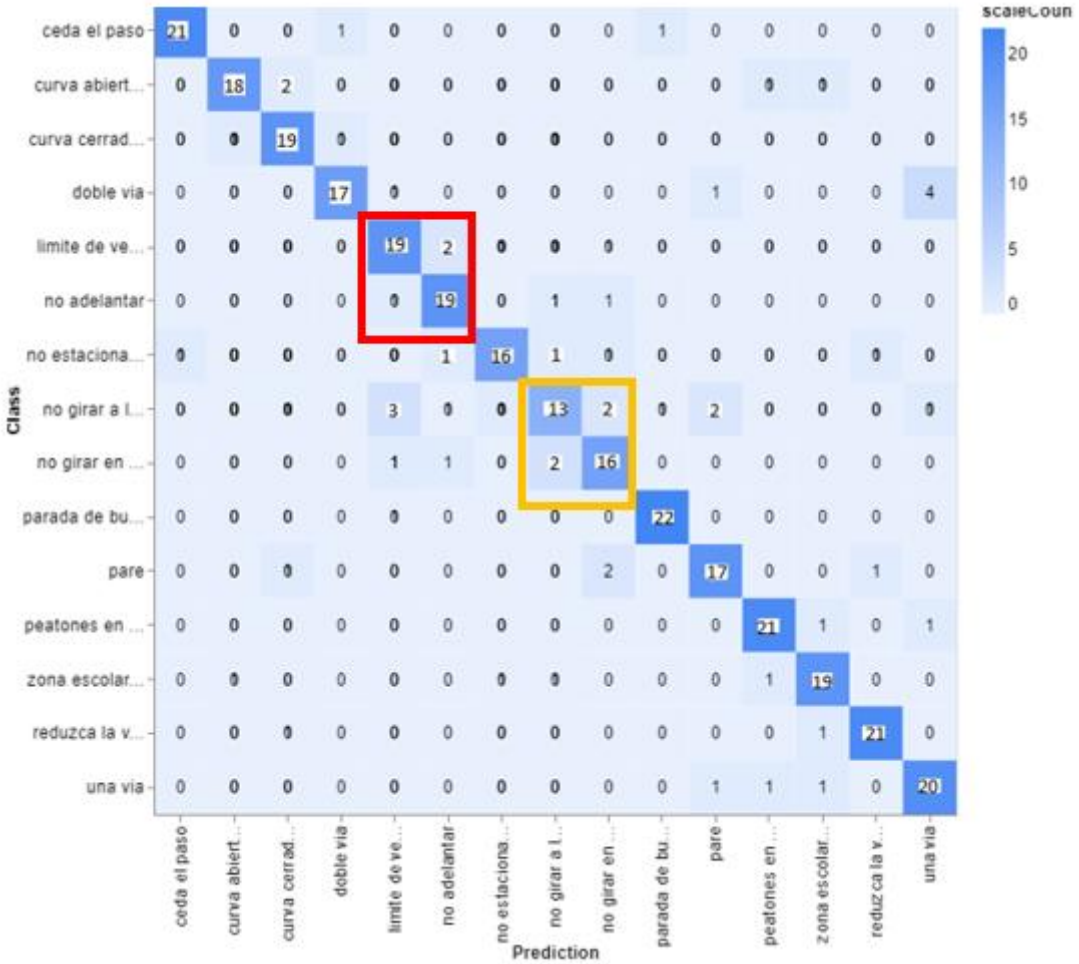
Parámetro	Época	Pérdida
Pérdida de clasificación (Loss)	100	0.0043
Pérdida de clasificación en el conjunto de pruebas (test acc)	100	0.9352

4.2.5. Matriz de confusión

Teachable Machine utiliza la matriz de confusión para mostrar de forma más detallada como el modelo clasifica las imágenes y con qué clase tiene más problemas para la detección, a su vez, con que clase se confunde durante el entrenamiento. La figura 44 presenta la matriz de confusión del modelo.

Figura 44

Matriz de confusión de Teachable Machine



En la figura 44 se ha señalado dos puntos en dónde se presenta una confusión entre 2 clases. Teachable Machine utiliza 23 imágenes de muestra para cada clase, además ha unificado las clases “curva cerrada a la derecha”, “curva cerrada a la izquierda”, “curvas cerradas a la derecha”, “curva cerrada a la izquierda”, “no girar a la derecha” y “no girar a la izquierda” en “curva abierta a la derecha e izquierda”, “curva cerrada a la izquierda y derecha” y “no girar a la derecha e izquierda”. Esto claramente es un problema, ya que indica que el modelo tiende a confundir estas clases debido a su similitud, en el recuadro de color naranja se observa que el modelo detecta 13 instancias correctas en la clase “no girar a la derecha e izquierda” teniendo una confusión de 3 instancias con la clase “límite de velocidad de 30km/h” y con 2 instancias en las clases “no girar en U” y “pare” respectivamente. Esto claramente es una deficiencia drástica en el modelo ya que, si el modelo no está completamente seguro de lo que ve en la imagen tiende a confundirlas aumentando el número de falsos positivos.

4.3. Inferencia en tiempo real

Para realizar la inferencia, se ha descartado los modelos Yolov8x y Teachable Machine debido a su poca eficiencia mostrada en los resultados de entrenamiento analizados anteriormente.

Se ha optado de dos formas parar realizar la inferencia, la primera es utilizando una computadora con sistema operativo Windows 10 con procesador 11th Gen Intel(R) Core (TM) i5-1135G7 y memoria RAM de 8 Gb. La segunda, mediante un Raspberry Pi 4 modelo B con un procesador quad-core AEM Cortex A72 a 1.5 Ghz y memoria RAM de 4 Gb. El objetivo es ver el rendimiento del modelo en dispositivos de gama media y como se adapta a dispositivos con límites de procesamiento y de qué manera se podría utilizar y sacar el mayor provecho al mismo.

4.3.1. Inferencia en computador

Para esta inferencia se utiliza la herramienta de PyCharm en donde se descargan las dependencias de yolov5 mostradas en la tabla 3. Cabe recalcar que la base de datos que se utilizó para el entrenamiento, está adaptada para detectar señales de tránsito en un ambiente real de conducción. Sin embargo, debido a los requerimientos de software y hardware que se necesita para llevarlo a un ambiente real, se optó por objetos que simulen señales de tránsito reales en un entorno de pruebas. En la figura 45, se observa la inferencia del modelo en tiempo real.

Figura 45

Inferencia de Yolov5n en dispositivo de gama media

Los resultados de la inferencia son aceptables, como se observa en la figura 45, la confiabilidad de cada detección no es inferior al 0.6 en su mayoría, lo que indica un buen rendimiento del modelo con pérdidas de confiabilidad moderadas.

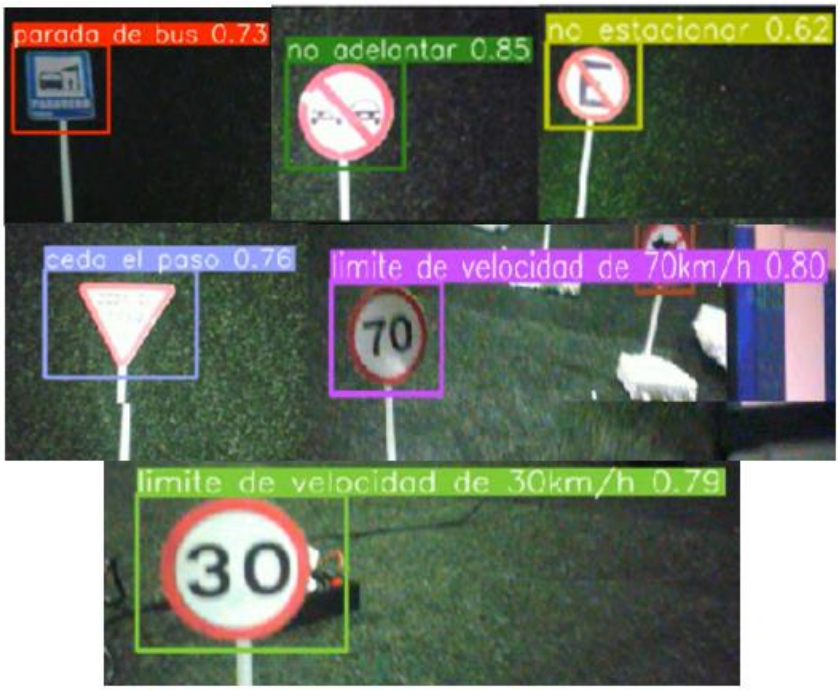
4.3.1. Inferencia en Raspberry Pi modelo 4

Existen dos opciones para utilizar el dispositivo Raspberry, de forma remota o directamente a través de la salida HDMI. Para este proyecto se utilizará la forma remota, para ello, se utiliza la aplicación RealVNC y la interfaz VNC del raspberry. La interfaz VNC se la habilita colocando el comando “sudo raspi-config”, seguido de ello, se escoge la opción interfaz y se habilita la interfaz VNC.

Para realizar esta inferencia es necesario utilizar entornos virtuales dentro la Respbebrry ya que permiten crear un entorno aislado con sus propias dependencias y bibliotecas, reduciendo conflictos de versiones específicas en las bibliotecas. Raspberry es un sistema embebido con limitaciones en procesamiento, debido a eso, la inferencia puede tener latencia en su detección. No obstante, es posible realizar detecciones en tiempo real con valores de confianza altos como muestra la figura 46.

Figura 46

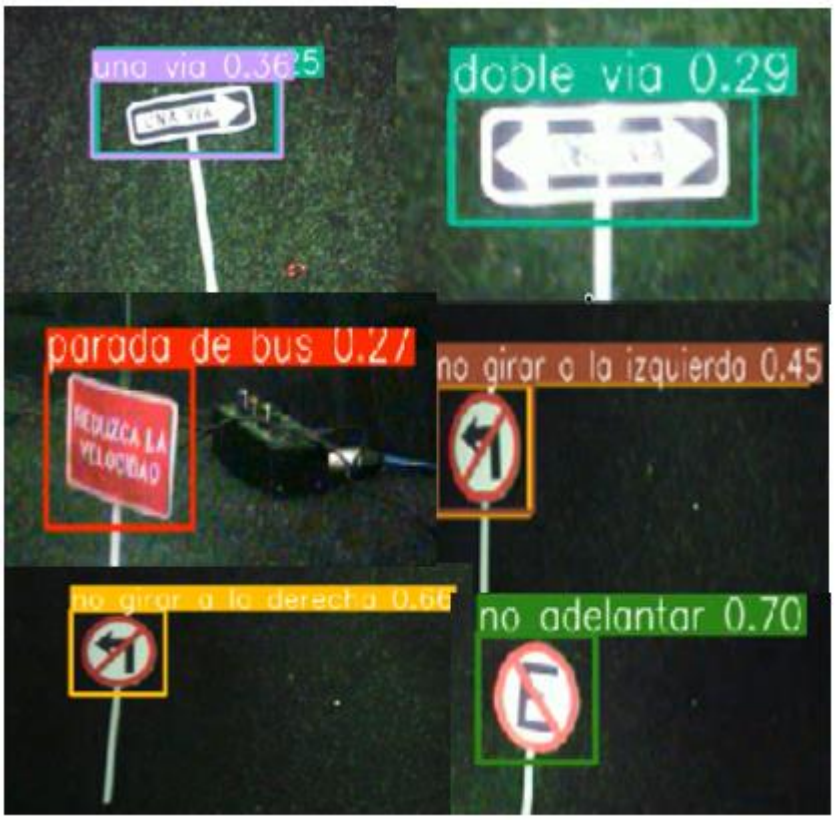
Inferencia en Rasberry Pi 4



Por otro lado, en la figura 47 se observa señales de tránsito en donde el modelo ha tenido dificultades en la detección, además se observa que el valor de confianza es relativamente bajo debido a las confusiones que el modelo tiene al detectar las señales de tránsito.

Figura 47

Dificultades de detección en Raspberry



Al analizar las figuras 46 y 47, se puede decir que la Raspberry no es el mejor dispositivo para implementar los modelos de detección basados en Yolo ya que necesita de requerimientos de procesamientos altos, otro dato particular es que las señales de tránsito detectadas deben estar cerca de la cámara del dispositivo, de lo contrario, el modelo empieza a cometer errores en la detección con más confusión en las señales con similitudes explicadas en los resultados anteriores.

Como una solución a este problema, se ha optado utilizar el modelo en la Raspberry para detectar las señales de tránsito por medio de imágenes y no en video. La figura 48 muestra los resultados de detección del modelo en la Raspberry con un conjunto de imágenes de señales de tránsito.

Figura 48

Inferencia con conjunto de imágenes en Raspberry

Como se observa en la figura 48, el rendimiento del modelo es aceptable, los valores de confiabilidad no son inferiores del 0.6. Utilizar el modelo con imágenes y no con video a resultado favorable ya que, para realizar las detecciones con video, se necesita una cámara con calidades de resolución altas. En la actualidad, existen múltiples cámaras con una excelente resolución, sin embargo, el problema no solo depende de la cámara. Para tener una fluidez grafica aceptable en raspberry se necesita entre 60 y 80 fps, Yolov5 alcanza los 150 fps en su

inferencia, es por ello, que la opción de implementar el modelo utilizando imagen de entrada, es la mejor opción para este proyecto.

CONCLUSIONES

Luego del análisis de técnicas y algoritmos de aprendizaje automático basados en detección de objetos dentro de una imagen, queda en evidencia que Yolov5 es la mejor opción para el modelo propuesto en el proyecto, debido a su método “You Only Look Once”, que permite realizar predicción en una sola pasada. La capacidad de Yolov5 para crear múltiples cuadros limitadores simultáneamente en tiempo real es especial para el modelo requerido, ya que garantiza una detección precisa y rápida del objeto.

El modelo resultante tiene la capacidad para detectar 20 señales de tránsito verticales del Ecuador con pérdidas de clasificación moderadas manteniendo un buen rendimiento con umbrales de confianza del 50 %. Es fundamental aclarar que, a pesar de que el modelo es estable, aun no se encuentra completamente listo para pruebas en entornos reales de conducción. Esto se debe a ciertas limitaciones en el aspecto de software y hardware. En el contexto de hardware, es crucial reconocer que la calidad de la detección de las señales de tránsito es directamente proporcional a la capacidad de la cámara utilizada.

A partir del análisis de mAP50-95, se concluye que el modelo mantiene un rendimiento consistente con valores no inferiores al 0.6. Aunque este valor puede considerarse relativamente bajo, es importante destacar que para esta medida se toma un umbral de confianza alto, en donde le modelo prioriza la precisión y confianza.

Raspberry es una computadora de bajo costo, sus funciones y características brindan un entorno favorable para tareas de detección de objetos mediante imágenes de entrada. Cabe recalcar que, para las inferencias en tiempo real, el dispositivo tiende a tener problemas de procesamiento debido a los fotogramas por segundo (fps).

RECOMENDACIONES

Para futuros entrenamientos, mejorar la base de datos sería primordial. La base de datos actual cuenta con 3016 imágenes con 20 clases diferentes, cada clase cuenta con 150 imágenes de las cuales 30 o 40 tienen variaciones de iluminación y similitudes entre ellas. Modificar esta cantidad de imágenes en la base de datos aportaría mejoras radicales al modelo.

Experimentar con más umbrales de confianza con el objetivo de encontrar un equilibrio entre precisión y tasa de detección. Este ajuste permite adaptar el modelo a las necesidades del proyecto, ya sea priorizando detecciones altamente confiables o maximizando la cobertura global de detecciones.

En el contexto de Raspberry, el utilizar GPU en lugar de CPU, aportaría mejoras para tareas de aprendizaje automático. Raspberry es una computadora de bajo costo y tamaño compacto con una CPU ARM que no puede ser tan potente como las CPU de las computadoras convencionales.

Optimizar el modelo es una opción para la implementación en la raspberry, teniendo en cuenta que unos de los problemas de la detección en el dispositivo era la cálida de la cámara que se utilizaba. Raspberry puede ser un potente procesador para modelos donde las detecciones no tengas tantas variedades en su base de datos.

Entrenar el modelo bajo diferentes épocas y batch, si bien cierto, un valor grande batch y épocas puede causar un sobre entrenamiento del modelo. Sin embargo, el probar con diferentes valores sin incrementarlos desmesuradamente, podría aportar con mejoras al modelo, Yolov5 trabaja con un número mínimo de 50 épocas dependiendo de lo que se quiera detectar. Por otro lado, el batch es número de imágenes que realiza para cada época, un valor alto del mismo provoca altas cargas computacionales, por ende, es necesario conocer el rendimiento del dispositivo donde se va realizar el entrenamiento.

ANEXOS

Anexo 1

Requerimientos de Yolov5

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt  # install

import torch
import utils
display = utils.notebook_init()  # checks
```

Anexo 2

Código de entrenamiento de Yolov5n

```
# Train YOLOv5s on data_señales for 100 epochs
!python train.py --img 640 --batch 32  --epochs 50 --data
/content/drive/MyDrive/data_señales/custon24.yaml --weights yolov5n.pt --
cache
```

Anexo 3

Requerimientos de Yolov8

```
%pip install ultralytics
import ultralytics
ultralytics.checks()
```

Anexo 4

Código de entrenamiento de Yolov8x

```
# Train YOLOv8n on COCO8 for 3 epochs
!yolo train model=yolov8x.pt
data=/content/drive/MyDrive/data_señales/custon24.yaml epochs=20
imgsz=640
```

Anexo 5

Código de Teachable Machine

```
from keras.models import load_model  # TensorFlow is required for Keras to
work
import cv2  # Install opencv-python
import numpy as np

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model =
load_model("C:/Users/MIGUEL/Desktop/tenserflow/modelo/keras_model.h5",
compile=False)

# Load the labels
class_names = open("C:/Users/MIGUEL/Desktop/tenserflow/modelo/labels.txt",
"r").readlines()

# CAMERA can be 0 or 1 based on default camera of your computer
camera = cv2.VideoCapture(1)

while True:
    # Grab the webcamera's image.
    ret, image = camera.read()

    # Resize the raw image into (224-height,224-width) pixels
    image = cv2.resize(image, (224, 224), interpolation=cv2.INTER_AREA)

    # Show the image in a window
    cv2.imshow("Webcam Image", image)

    # Make the image a numpy array and reshape it to the models input shape.
    image = np.asarray(image, dtype=np.float32).reshape(1, 224, 224, 3)

    # Normalize the image array
    image = (image / 127.5) - 1

    # Predicts the model
    prediction = model.predict(image)
    index = np.argmax(prediction)
    class_name = class_names[index]
    confidence_score = prediction[0][index]

    # Print prediction and confidence score
    print("Class:", class_name[2:], end="")
    print("Confidence Score:", str(np.round(confidence_score * 100))[:-2],
"%")

    # Listen to the keyboard for presses.
    keyboard_input = cv2.waitKey(1)

    # 27 is the ASCII for the esc key on your keyboard.
    if keyboard_input == 27:
        break
```

```
camera.release()
cv2.destroyAllWindows()
```

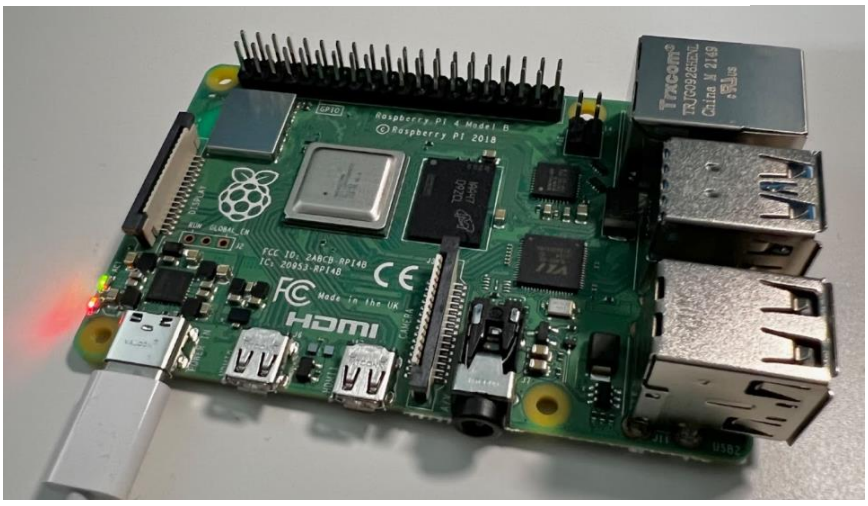
Anexo *6*

Código de implementación de Yolov5n en PyCharm

```
import torch
import cv2
import numpy as np

# Leemos el modelo
model = torch.hub.load('ultralytics/yolov5', 'custom',

path='C:/Users/MIGUEL/Desktop/yolov5/yolov5n/yolov5n_100ep.pt')

# Video captura
cap = cv2.VideoCapture(1)

# Empezamos
while True:
    # Realizar la lectura del video
    ret, frame = cap.read()

    # Realizamos la detección
    detect = model(frame)

    # Mostramos fps
    cv2.imshow('detector de señales', np.squeeze(detect.render()))

    # Leer teclado
    key = cv2.waitKey(5)
    if key == 27:
        break  # ¡La declaración 'break' está dentro del bucle 'while'!

cap.release()
cv2.destroyAllWindows()
```

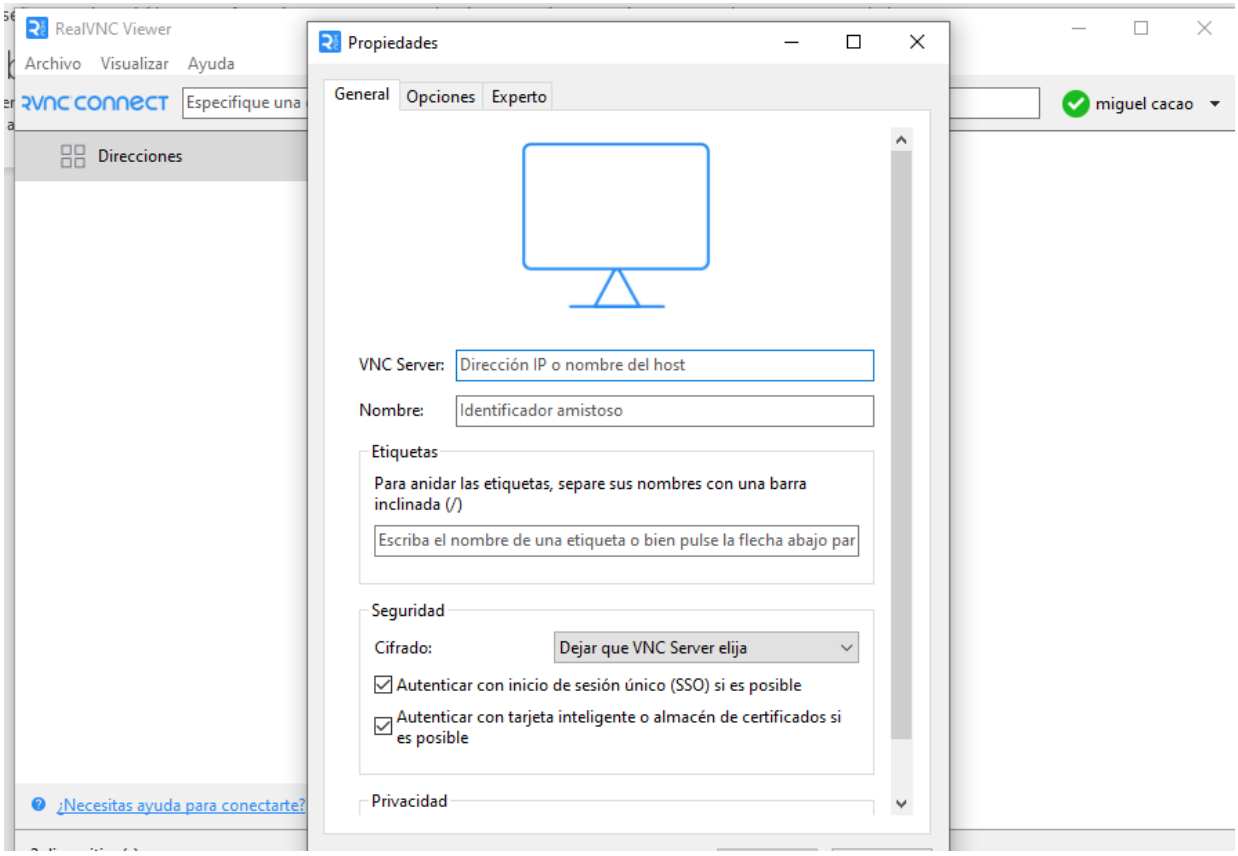
Anexo 7

Dispositivo Raspberry



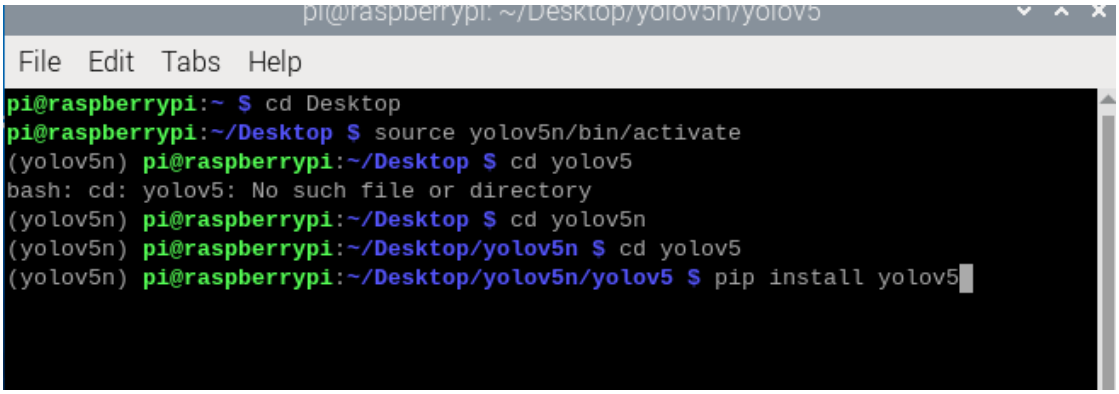
Anexo 8

Aplicación para conexión remota a la Raspberry



Anexo 9

Instalación de requerimientos de Yolov5 en Raspberry



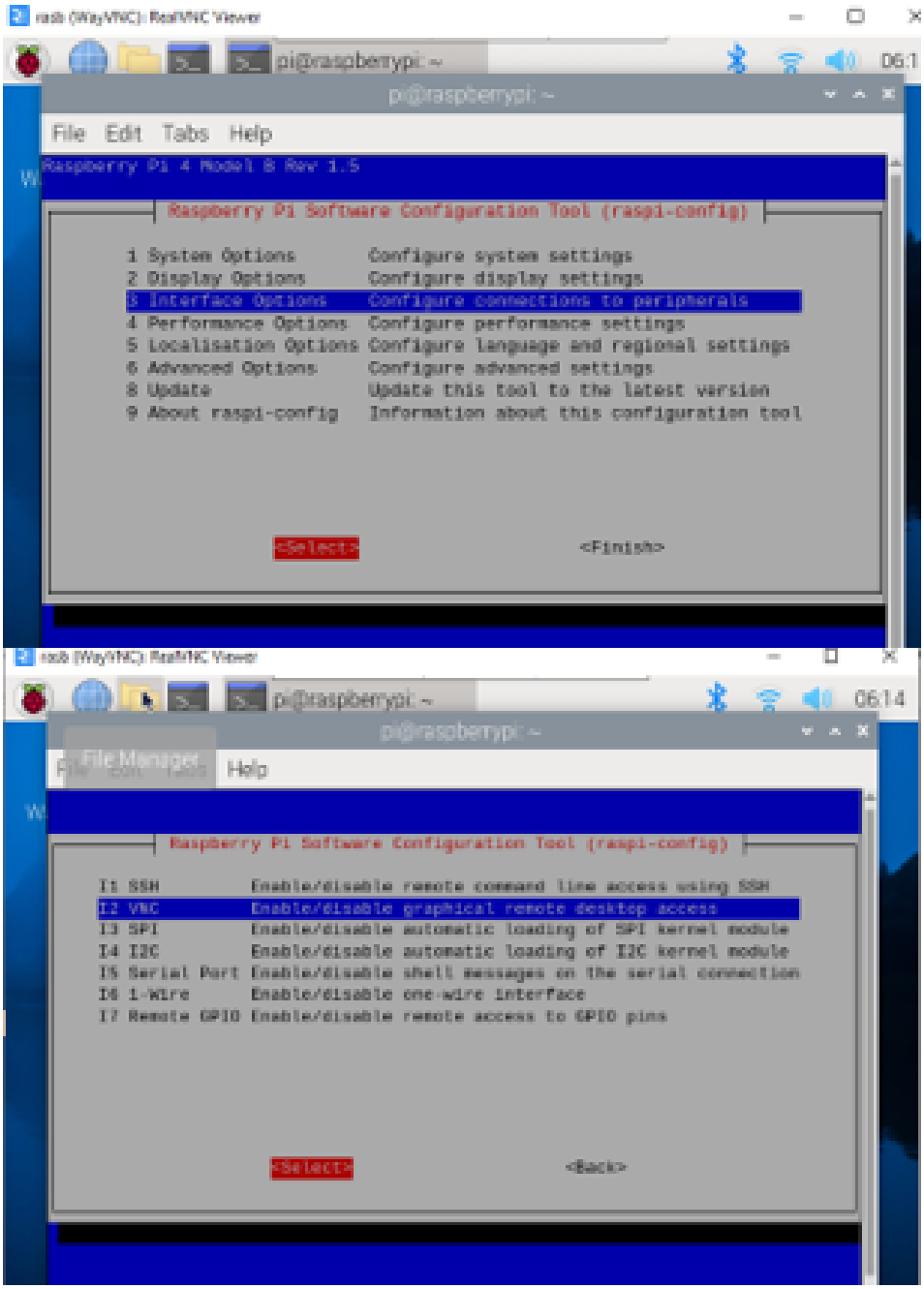
Anexo 10

Código de implementación de Yolov5 en Raspberry

```
import torch
import cv2
import numpy as np
import psutil  # Importa la biblioteca psutil
# Leemos el modelo
model = torch.hub.load('ultralytics/yolov5', 'custom',
                       path='/home/pi/Desktop/yolov5n/yolov5/yolov5n_100ep.pt')
# Video captura
cap = cv2.VideoCapture(0)
# Empezamos
while True:
    # Realizar la lectura del video
    ret, frame = cap.read()
    # Realizamos la deteccin
    detect = model(frame)
    # Mostrar fps
    cv2.imshow('detector de senales', np.squeeze(detect.render()))
    # Monitorear el uso de la memoria RAM
    ram = psutil.virtual_memory()
    print(f"Porcentaje de uso de la memoria RAM: {ram.percent}%")
    # Leer teclado
    key = cv2.waitKey(5)
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Anexo 11

Activación de VNC



BIBLIOGRAFÍAS

A Cordon Ureña. (2021). *ALEJANDRO CORDÓN UREÑA 81020_ALEJANDRO_CORDON_URENA_Aprendizaje_profundo_en_IoT_Redes_neuronales_convolucionales_con_imagenes_aplicadas_a_un_vehiculo_autonomo_FINAL_2_1443006176*.

Agua, L., Fabian, C., Fausto, M. I., & Bravo, A. R. (2014). *UNIVERSIDAD CENTRAL DEL ECUADOR FACULTAD DE INGENIERÍA CIENCIAS FÍSICAS Y MATEMÁTICA CARRERA DE INGENIERÍA CIVIL DIAGNOSTICO DEL PROBLEMA DE CONGESTIÓN VEHICULAR Y*.

Ameijeiras Sánchez, D., & Hernández Heredia, Y. (2020). Revisión de algoritmos de detección y seguimiento de objetos con redes profundas para videovigilancia inteligente Algorithms for detection and tracking objects with deep networks for intelligent video surveillance: A review. *Revista Cubana de Ciencias Informáticas*, *14*(3). http://rcci.uci.cuPág.165-195Editorial"EdicionesFuturo"http://orcid.org/0000-0002-6734-7493HéctorR.GonzálezDiez1http://orcid.org/0000-0002-7601-4201http://orcid.org/0000-0001-9433-5511

Aranda Barjola. (2012). *UNIVERSIDAD CARLOS III DE MADRID ESCUELA POLITÉCNICA SUPERIOR MEDIANTE CÁMARA EMBARCADA*.

Artola Moreno Tutor, Á., & Antonio Pérez Carrasco, J. (2019). *Clasificación de imágenes usando redes neuronales convolucionales en Python*. https://idus.us.es/handle/11441/89506

Blacio, M., 21, J. S.-… páginas;, 29, 0 x, 7, undefined, & 2021, undefined. (2021). SISTEMA INTEGRAL DE HOGAR INTELIGENTE BASADO EN HOME ASSISTANT Y RASPBERRY PI. *Researchgate.NetMAM Blacio, JEB Sarmiento, OGJ Sarango, EEG MallaPaúl Baldeón Egas, Grisel Pérez, Julio C. Riascos, Brigitte González, 2021•researchgate.Net*. https://www.researchgate.net/profile/Maryory-Urdaneta-2/publication/353826882_Sistema_de_seguimiento_de_requerimientos_eventos_e_incidentes_para_los_clientes_de_la_empresa_TELCONET_SA_en_la_ciudad_de_Quito/links/6113d39b1e95fe241ac5c3d5/Sistema-de-seguimiento-de-requerimientos-eventos-e-incidentes-para-los-clientes-de-la-empresa-TELCONET-SA-en-la-ciudad-de-Quito.pdf#page=101

Buitrago, J., Vera, M., & Niño, J. (n.d.). *AUTOS Y SU INDEPENDIZACI´ONINDEPENDIZACI´ INDEPENDIZACI´ON*.

Carolina Armendariz Rodríguez, V. I., Geovanny Chandi Enriquez, A. I., & Javier Gallegos Nogales, L. I. (2023). *Ciencias de la Educación Artículo de Investigación*. https://doi.org/10.23857/pc.v8i7

Catapang, A., International, M. R.-2016 6th I., & 2016, undefined. (2016). *Obstacle detection using a 2D LIDAR system for an Autonomous Vehicle*. https://ieeexplore.ieee.org/abstract/document/7893614/

Cerpas Alonso. (2021). *fast_RCNN*.

Chicaiza, F. S. (2017). *Sistema electrónico de alerta automática para el reconocimiento de señales de tránsito reglamentarias, preventivas e informativas en la Ciudad Ambato*. http://repositorio.uta.edu.ec/handle/123456789/26945

Durán Suárez, J., Del, A., Torres, R., & Suárez, D. (2017). *Redes neuronales convolucionales en R: Reconocimiento de caracteres escritos a mano*. https://idus.us.es/handle/11441/69564

Flores-Calero, M., Conlago, C., Yunda, J., Aldás, M., & Flores, C. (2018). *Artículo Científico / Scientific Paper Implementación de un algoritmo para la detección de señales de tránsito del Ecuador: Pare, Ceda el paso y Velocidad Implementation of an algorithm for Ecuadorian traffic sign detection: Stop, Give-way and Velocity cases*. https://doi.org/10.17163/ings.n20.2018.01

Gómez Flores, W. (2015). *Reconocimiento de objetos en fotografias*. https://www.tamps.cinvestav.mx/~wgomez/toptamps/tutorial.pdf

González-García, C. (n.d.). *En qué consiste el aprendizaje automático (machine learning) y qué está aportando a la Neurociencia Cognitiva*. Retrieved November 16, 2023, from www.cienciacognitiva.org

Irimia, M. C. (2021). *Desarrollo de módulos de visión por computador en Python para la detección de objetos en un entorno de pruebas de conducción autónoma*. https://ruc.udc.es/dspace/handle/2183/30487

Javier, A., & Vargas, A. (2022). *Detection of vertical traffic signaling with convolutional neural networks based on residual blocks*. https://orcid.org/0000-0002-4716-4112

Jhan, G., & Arteaga, P. R. (2015). *APLICACIÓN DEL APRENDIZAJE PROFUNDO ("DEEP LEARNING") AL PROCESAMIENTO DE SEÑALES DIGITALES UNIVERSIDAD AUTÓNOMA DE OCCIDENTE FACULTAD DE INGENIERÍA DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA PROGRAMA DE INGENIERÍA MECATRÓNICA SANTIAGO DE CALI 2015*.

Li, J., Li, C., Fei, S., Shi, J., Xiao, Z., Ma, C., Chen, W., Ding, F., Wang, Y., & Li, Y. (2021). Wheat Ear Recognition Based on RetinaNet and Transfer Learning. *Sensors 2021, Vol. 21, Page 4845*, *21*(14), 4845. https://doi.org/10.3390/S21144845

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9905 LNCS*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2/FIGURES/5

López, R., & Fernández, J. (2008). *Las redes neuronales artificiales*. https://books.google.cl/books?hl=es&lr=&id=X0uLwi1Ap4QC&oi=fnd&pg=PA11&dq=%5B22%5D%09L%C3%B3pez,+R.+F.,+%26+Fern%C3%A1ndez,+J.+M.+F.+(2008).+Las+redes+neuronales+artificiales.+Netbiblo.&ots=gPHwhooq-k&sig=a9ifl9MVhI-gPHON8HokT1e24Wo

MARTIN ARCE CIGÜEÑAS. (2017). *PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ*.

Millán, K. A., & Alzate, I. V. (2018). *Reconocimiento de patrones en imágenes de video para el monitoreo de eventos de tráfico vehicular en Santiago de Cali*. https://red.uao.edu.co/bitstream/handle/10614/10190/T07853.pdf?sequence=5

Montaño, M., Torres, R., Ludeña, P., & Sandoval, F. (2021). IoT Management Analysis Using SDN: Survey. *Communications in Computer and Information Science*, *1388 CCIS*, 574–589. https://doi.org/10.1007/978-3-030-71503-8_45/COVER

Montaño-Blacio, M., Briceño-Sarmiento, J., & Pesántez-Bravo, F. (2021). 5G Network Security for IoT Implementation: A Systematic Literature Review. *Advances in Intelligent Systems and Computing*, *1277*, 28–40. https://doi.org/10.1007/978-3-030-60467-7_3

Nelson Morales. (2015). *Investigación Exploratoria: Tipos, Metodología y Ejemplos*.

Palacios, V. M. (2015). *Diseño de un controlador de velocidad, homologado a los límites de velocidad de Ecuador*. http://repositorio.ug.edu.ec/handle/redug/17275

Purwita Sary, I., Ucok Armin, E., Andromeda, S., Engineering, E., & Singaperbangsa Karawang, U. (2023). Performance Comparison of YOLOv5 and YOLOv8 Architectures in Human Detection Using Aerial Images. *Ultima Computing : Jurnal Sistem Komputer*, *15*(1).

REY, L., & JUAN, V. (2021). *RECONOCIMIENTO DE SEÑALES DE TRÁFICO MEDIANTE VISIÓN ARTIFICIAL Y REDES NEURONALES*. https://www.uv.es/lapeva/Thesis/TFG_2021_Laura_Garrido_compressed.pdf

Rosa Gonzales, Javier Machacuay, Pedro Rotta, & César Chingel. (2022). *Hyperparameters Tuning of Faster R-CNN Deep Learning Transfer for Persistent Object Detection in Radar Images* (Vol. 20, Issue 4).

Sanchez Vidal. (2021). *Lara Sanchez Vidal*.

Saúl, D., & Raneros, R. (2021). *Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning*. https://uvadoc.uva.es/handle/10324/45359

Vial, E., & Vial, S. (2015). *La educación vial en Primaria*. https://uvadoc.uva.es/bitstream/handle/10324/14649/TFG-G1426.pdf?sequence=1