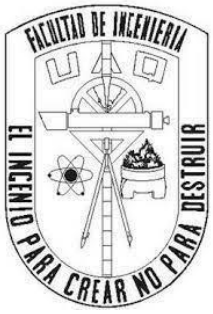

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

CAMPUS SAN JUAN DEL RÍO

FACULTAD DE INGENIERÍA

INGENIERÍA MECÁNICA Y AUTOMOTRIZ

“SISTEMA DE RECONOCIMIENTO DE SEÑALES MEXICANAS DE TRÁNSITO PREVENTIVAS MEDIANTE APRENDIZAJE DE MÁQUINAS”

PRESENTA:

JOSÉ AMAURY ARELLANO JIMÉNEZ

DIRIGIDO POR:

DR. MARTÍN VALTIERRA RODRÍGUEZ

SAN JUAN DEL RÍO, QUERÉTARO, 2023

SISTEMA DE RECONOCIMIENTO DE SEÑALES MEXICANAS DE TRÁNSITO PREVENTIVAS MEDIANTE APRENDIZAJE DE MÁQUINAS

por

José Amaury Arellano Jiménez

RESUMEN

El presente trabajo de tesis se centró en la implementación de algoritmos de detección de objetos y reconocimiento de imágenes mediante aprendizaje de máquinas con el objetivo de desarrollar un sistema de detección y reconocimiento de señales mexicanas de tránsito preventivas. Para el proceso de entrenamiento de los modelos de detección y clasificación se recolectaron un total de 8665 imágenes de distintas señales de tráfico mexicanas. En este trabajo se hizo uso de dos técnicas de aprendizaje de máquinas para la detección de objetos: redes neuronales convolucionales y clasificadores en cascada. Se obtuvo un modelo de detección de señales de tráfico a partir del entrenamiento de una red neuronal convolucional YOLOv5x. El segundo modelo de detección se desarrolló partiendo de la técnica conocida como Haar Cascade (Cascadas de Haar). Para el modelo de reconocimiento se diseñó y entrenó una red neuronal convolucional para la clasificación de imágenes. Para este trabajo se seleccionaron cinco clases de señales mexicanas de tránsito preventivas: curva a la derecha, curva a la izquierda, curva sinuosa, límite de altura y restrictiva. La programación del sistema de reconocimiento se llevó a cabo en lenguaje Python haciendo uso de las librerías OpenCV, TensorFlow y Keras. Se aplicó en el sistema de detección y reconocimiento de señales de tránsito una matriz con 40 pruebas por clase. Las pruebas se llevaron a cabo bajo condiciones nominales de clima, iluminación y nivel de desgaste en los señalamientos sometidos a estudio. El modelo de detección de Haar Cascade mostró mayor velocidad de procesamiento en la detección de señalamientos de tránsito en comparación con el modelo YOLOv5x. El modelo de clasificación de señales de tráfico demostró reconocer las señales seleccionadas para el estudio con una tasa de exactitud superior al 95%.

Palabras clave: Redes neuronales convolucionales, YOLOv5, Haar Cascade, señales de tránsito, Python, OpenCV.

ABSTRACT

The present thesis work focused on algorithms of object detection and image recognition through machine learning with the aim of developing a system of detection and recognition of Mexican preventive traffic signals. For the training process of the detection and classification models, a total of 8665 images of different Mexican traffic signs were collected. In this work, two machine learning techniques for object detection were used: Convolutional neural networks and cascade classifiers. A traffic signal detection model was obtained from the training of a convolutional neural network YOLOv5x. The second detection model was developed based on the technique known as Haar Cascade. For the Recognition model, a convolutional neural network for image classification was designed and trained. For this work, five classes of Mexican preventive traffic signals were selected: right curve, left curve, sinuous curve, height limit and restrictive. The programming of the recognition system was carried out in Python language using the OpenCV, TensorFlow and Keras libraries. A matrix with 40 tests per class was applied in the detection and recognition system of traffic signals. The tests were carried out under nominal conditions of weather, lighting, and wear level in the signs under study. The Haar Cascade detection model showed higher processing speed in traffic signal detection compared to the YOLOv5x model. The traffic signal classification model demonstrated to recognize the signals selected for the study with an accuracy rate of more than 95%.

Keywords: Convolutional neural networks, YOLOv5, Haar Cascade, traffic signals, Python, OpenCV.

DEDICATORIA

Este trabajo va dedicado especialmente a mis padres por siempre brindarme atención, cariño y apoyo a largo de toda mi trayectoria académica, pero sobre todo a lo largo de mi vida. Este trabajo es el resultado de todo el esfuerzo que ellos han hecho al siempre brindarme las mejores oportunidades y experiencias que estuvieron a su alcance para mi crecimiento. Soy consciente de que no me alcanza esta vida ni la que sigue para terminar de agradecerles todo lo que han hecho por mí.

A mis abuelos por todo el cariño y apoyo brindado a lo largo de mi vida, siempre los llevo conmigo. Y aunque uno ya no está conmigo para compartir este logro se lo dedico hasta el cielo. Ya cumplí una de las dos promesas que hice el día que se fue, solo le pido a la vida la oportunidad de cumplir la que me falta.

A mi hermano por ser esa persona en la que puedo confiar sin importar nada.

AGRADECIMIENTOS

Quiero agradecer a mis padres por todos los sacrificios que tuvieron que hacer a lo largo de mi vida para que yo alcanzara esta meta. Todo lo que soy y lo que tengo es gracias a ellos. Quiero agradecerles por formar en mí una persona de valores y de una ética inquebrantable.

De manera individual quiero agradecer a mi mamá por siempre estar presente en mi desarrollo, por ser mi confidente y por brindarme su apoyo en los momentos más difíciles. Quiero decirle que todas esas veces que tuvo que disfrazarse y bailar en festivales escolares valieron la pena y hoy puede decir que su hijo se convirtió en ingeniero.

A mi papá por brindarme las oportunidades a las que él no tuvo acceso, por su protección y cariño. Quiero agradecerle por siempre estar presente y apoyarme en todo. Pero sobre todo quiero agradecerle por ser mi guía y más grande maestro en la vida. Siempre supe que tipo de hombre quería ser en la vida viéndolo a él.

Quiero agradecer a mis compañeros porque en ellos encontré personas extraordinarias que se convirtieron en mis amigos y hoy sin duda los considero mis hermanos Rubén, Cristian, Saúl, Sergio, Héctor y Diego . Sin dudarlo me apoyaron en los momentos más difíciles de la carrera.

A mi director de tesis el Dr. Martín Valtierra Rodríguez por darme la oportunidad de realizar este proyecto con él, por siempre estar pendiente, por el seguimiento que le dio a este proyecto, pero sobre todo por sus enseñanzas y su amistad.

A la Universidad Autónoma de Querétaro por darme una formación académica de calidad. Por el prestigio de pertenecer a esta casa de estudios y el orgullo de ser 100% universitario.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1.- Introducción.

En el sector automotriz la visión artificial ha cobrado gran importancia en los últimos 15 años con trabajos de investigación y de desarrollo tecnológico enfocados a vehículos autónomos y de asistencia a la conducción. Dentro de las aplicaciones de asistencia a la conducción se encuentra la detección y reconocimiento de señales de tránsito, este tipo de sistemas tienen el objetivo de alertar al conductor de las señalizaciones de tránsito que se presentan durante el seguimiento del camino o incluso ejercer acciones de reducción de velocidad en caso de ignorar por completo las señales restrictivas detectadas. Marcas comerciales de vehículos han implementado este tipo de tecnología en sus modelos de autos. Por ejemplo, Tesla (2022) ha desarrollado un sistema de control de semáforos y señales de stop (alto) utilizando visión artificial. Esta función de asistencia es capaz de mostrar mensajes de emergencia cuando se detecta un semáforo, señal de alto o una señal de tráfico, además de tener la capacidad de reducir la velocidad en cruces viales, e incluso puede detener por completo el vehículo si se detecta un semáforo en rojo. Estos sistemas basan su funcionamiento en lo que se conoce como aprendizaje automático el cual es un conjunto de algoritmos que resuelve problemas a partir de la toma de decisiones las cuales se guían por la experiencia acumulada (Moreno et al.,1994).

El proyecto de tesis se centra en la detección y reconocimiento de señales de tránsito preventivas mexicanas utilizando visión artificial. El interés por investigar y desarrollar este tema en específico nace del hecho de que actualmente la industria automotriz que se enfoca cada vez más en la movilidad autónoma, por lo que el primer paso es la creación de aplicaciones para la asistencia a la conducción. Otra motivación para llevar a cabo este proyecto es el de sentar las bases de investigación y de desarrollo tecnológico que en un futuro sirvan para crear un sistema alternativo de detección de señales de tránsito completamente funcional, dirigido aquellos vehículos del parque vehicular mexicano que no cuentan con sistemas que ofrezcan asistencia al momento de conducir, esto con el objetivo de

tratar de evitar lesiones por incidentes de tráfico y muertes por ignorar o desobedecer señalamientos viales preventivos.

1.2.- Antecedentes

Se han reportado escritos de investigación que utilizan distintas técnicas de detección y reconocimiento de señalizaciones viales de los que destacan artículos de congreso, artículos de revistas de divulgación científica, así como trabajos de tesis para la obtención de grado. Los documentos presentados basan sus metodologías a partir de algoritmos que trabajan con aprendizaje de máquinas (del inglés machine learning) o aprendizaje profundo (del inglés deep learning), siendo el segundo una variante del primero.

En el plano internacional Farag (2018) plantea el desarrollo de un clasificador partiendo del uso de una red neuronal convolucional CNN (del inglés Convolutional Neural Network) conocida como WAF-LeNet, la cual tiene una composición de 15 capas y es utilizada para identificación de señales de tráfico. Para la etapa de entrenamiento de la CNN se utilizó el algoritmo de optimización Adam además de utilizar la base de datos de señales de tráfico alemanas (del inglés German Traffic Sign Dataset – GTSRB), con dicho algoritmo el autor alcanza un porcentaje de exactitud en las sesiones de prueba del 96.5% en la identificación de señales de tránsito. William et al. (2019) proponen un modelo para la detección y reconocimiento de señales de tránsito en tiempo real considerando los diferentes ambientes de iluminación, visibilidad y condiciones climáticas. Los autores utilizan un método de aprendizaje de transferencia y se basan en el estado del arte de los sistemas de detección multi-objeto con una red neuronal de convolución basada en regiones más rápidas F-RCNN (del inglés Faster Region-based Convolutional Neural Network), un detector multi-objeto de un solo disparo SSD (del inglés Single Shot Detector) y la combinación de dos extractores de características: Inception v2 y Tiny-Yolo v2 con los cuales generan dos modelos de prueba. Los resultados reportados por los autores muestran que el modelo F-RCNN Inception v2 tiene un mejor rendimiento en condiciones adversas reales además de ser más rápido, exacto y fiable con una media de exactitud del 96% mientras que el modelo Tiny-

YOLO v2 si bien tiene resultados decentes los autores recomiendan el uso de los modelos YOLOv2 o YOLOv3 si se busca una exactitud más alta. Ayachi et al. (2019) utilizando la técnica de aprendizaje profundo para procesamiento de imágenes basado en redes neuronales de convolución desarrollan una aplicación robusta para la detectar señalamientos de tráfico. La aplicación es implementada en un sistema embebido. Los escritores del artículo reportan una exactitud media de detección del 84.22%. Sun et al. (2019) desarrollan una técnica para reconocimiento de señales de tránsito mediante Deep Learning, la técnica presentada tiene el objetivo de detectar y clasificar señales de tipo circular. El método se basa en aplicar un preprocesamiento a la imagen para extraer sus características más importantes de la misma posteriormente se aplica la Transformada de Hough para detectar líneas y bordes en la imagen, después utilizando una CNN se realiza la clasificación de las imágenes que contienen señales de tránsito circulares. En el artículo se reporta una tasa de acierto del 98.2%. Novak et al. (2020) presentan un software basado en una red YOLOv3 conectada a una CNN el cual es entrenado para el reconocimiento de señalamientos viales. La red YOLOv3 fue previamente entrenada para detectar y clasificar cinco tipos de objetos: coches, camiones, peatones, señales de tráfico y semáforos. Cuando la red YOLOv3 detecta una señal de tráfico transmite esta señal a la red CNN, esta red es capaz de clasificar la señal de tránsito dentro de una de las 75 categorías con las que fue entrenada. En el artículo se reporta un alto nivel de confianza en el algoritmo de reconocimiento con el 99.2% de exactitud. Abdi y Medded (2017) proponen un sistema de detección y reconocimiento de señales de tráfico combinando técnicas de cascada y deep learning en tiempo real. Los autores reportan que la combinación de Haar Cascade con redes neuronales profundas mejora notablemente el desempeño de detección sin perder rendimiento en tiempo real.

Dentro de la industria automotriz algunas marcas ya han adoptado este tipo de sistemas los cuales se incorporan en algunos de sus modelos de venta comercial. Ford (2022) cuenta con una función denominada “Sistema de reconocimiento de señales de tráfico”, está función tiene la capacidad de detectar y reconocer señales de tránsito fijas o transitorias que se encuentren a un costado del

camino. Renault (2022) también cuenta con una tecnología similar a la que nombra “Detección de paneles de señalización”, su sistema es capaz de reconocer señales de límite de velocidad y de mostrar el límite detectado en el panel de instrumentos, además de advertir zonas de prohibición para adelantamientos. Si se excede el límite de velocidad en el tablero de instrumentos el icono de la señal detectada comienza a parpadear, también se le suma una señal sonora al mensaje de advertencia. Mercedes-Benz (2022) incorpora un sistema conocido como “Asistente para señales de tráfico” esta función además de reconocer límites de velocidad y prohibiciones para el adelantamiento de vehículos también cuenta con mensajes de advertencia si el vehículo conduce sobre una vía en sentido contrario.

A nivel nacional Rodríguez et al. (2020) presentan un artículo en el que realizan la detección y clasificación de señales de tráfico mexicanas mediante aprendizaje profundo. En este trabajo los autores entrenan una red neuronal de convolución utilizando la colección de imágenes CIFAR-10 la cual es ampliamente utilizada para el entrenamiento de algoritmos de visón por computadora. Posteriormente realizan una transferencia de aprendizaje de la CNN entrenada con CIFAR-10 a una red neuronal convolucional basada en regiones R-CNN (del inglés Region-based Convolutional Neural Network), esto con el fin de realizar la detección de las señales de tráfico. Por último, los autores hacen uso de una CNN previamente entrenada conocida como ResNet-50 a la que se le modificaron sus tres últimas capas las cuales realizan la clasificación de las distintas señales de tránsito. A esta CNN modificada se le realiza una transferencia de aprendizaje de la R-CNN para poder detectar y clasificar señalamientos viales. Los autores logran una tasa de exactitud de reconocimiento del 95.33%. Por su parte de Jesús (2020) realizó un sistema de detección de señales de tránsito a partir de redes neuronales de convolución con pre-entrenamiento en la detección de objetos conocidas como YOLOv2. En el estudio se compararon tres redes neuronales: una red YOLOv2 en su estado original, una YOLOv2 modificada la cual el autor denominó Imp. YOLOv2. Las modificaciones aplicadas a dicha red fueron la implementación de una función de perdida para eliminar términos redundantes y apegarse más a las etiquetas de los datos además de modificar los valores para los cuadros de anclaje (del inglés

anchors boxes) con el objetivo de incrementar la exactitud del algoritmo. La tercera red neuronal comparada fue una red del tipo F-RCNN. De acuerdo con los resultados reportados por el autor la red Imp. YOLOv2 tiene un desempeño superior a sus contrapartes comparadas en el estudio siendo mejor en la detección de señales de tránsito en imágenes que presentan cambios en el clima, así como condiciones de desgaste en los señalamientos.

De manera local en el repositorio institucional de la Universidad Autónoma de Querétaro existe un proyecto de tesis de maestría que presenta un algoritmo para la detección de vehículos y peatones el cual utiliza redes neuronales de convolución. En este proyecto Treviño (2022) mediante la combinación de dos técnicas de inteligencia artificial utilizadas para la detección y clasificación de objetos: redes neuronales de convolución y los algoritmos de búsqueda propone un modelo hibrido de estas dos técnicas con el objetivo de mejorar el proceso de identificación de una CNN al implementar un algoritmo metaheurístico. El autor reporta en sus resultados que al implementar algoritmos metaheurísticos para el ajuste fino de los parámetros de una red YOLOv3 se puede notar una mejoría en la detección de vehículos debido a que el algoritmo mejora la precisión global en la detección de diversas clases. Por su parte Alcántara (2018) presenta otra tesis de maestría cuyo objetivo es el desarrollo de un algoritmo de detección de somnolencia en conductores a través de un detector facial basado en descriptores HOG y clasificadores de cascada. A través de las expresiones faciales en conductores el algoritmo determina la presencia de somnolencia en el usuario. Si el sistema detecta presencia de somnolencia en el conductor se activa un sistema de alarma.

De acuerdo con la revisión de los antecedentes del tema, los trabajos revisados abordan de alguna manera la detección y reconocimiento de señales de tráfico a través del uso de técnicas de machine learning o deep learning, sin embargo, los autores aún no han reportado haber realizado pruebas en línea, es decir en un vehículo que viaje a diferentes velocidades. En este sentido, el presente trabajo reportará los resultados que se obtengan con una base de datos propia de señales de tráfico mexicanas respecto al tiempo de cómputo de la detección y

clasificación de estos señalamientos considerando los cambios asociados en las imágenes debido a la velocidad del vehículo.

1.3.- Descripción del problema

Desarrollar sistemas de detección y reconocimiento de señales de tráfico representa un gran reto esto es debido a que el investigador del tema se enfrenta a diferentes problemáticas. Por ejemplo, cuando las condiciones de iluminación son deficientes los sistemas de reconocimiento tienen dificultades para poder detectar y clasificar las señales de tránsito. Las condiciones climáticas también juegan un papel importante a la hora de intentar detectar y reconocer señalamientos viales, condiciones como lluvia, neblina e incluso nieve generan adversidades al momento de realizar tareas de reconocimiento y clasificación de este tipo de señalizaciones. Otra problemática que se presenta al momento de intentar detectar y reconocer señales viales en tiempo real es el desenfoque de la cámara, esto debido a la velocidad y movimiento del vehículo lo cual genera imágenes deficientes para que trabaje el algoritmo de reconocimiento. El tiempo de procesamiento del sistema es otro problema para considerar ya que las aplicaciones en tiempo real para este tipo de sistemas requieren tiempos de procesamiento muy cortos. La falta de un conjunto robusto de datos que contenga señales de tránsito mexicanas dificulta la tarea de desarrollar un sistema de detección y reconocimiento que funcione de forma eficiente en carreteras de México. Todo lo anterior genera la necesidad de desarrollar modelos que tengan la capacidad de ser rápidos y precisos al momento de detectar y clasificar los distintos señalamientos de tránsito en México.

Este trabajo de tesis está centrado en resolver la problemática que se tiene cuando se intenta detectar y reconocer señales de tráfico a diferentes velocidades de viaje. La carga computacional del sistema de detección y clasificación de señales de tránsito es otro problema que se buscará resolver en este proyecto. Además de desarrollar un sistema de detección y reconocimiento de señales de tráfico mexicanas, este trabajo de tesis también buscará contribuir con la creación de una base de datos que contenga señales mexicanas de tránsito preventivas y restrictivas para dar soporte a futuras investigaciones.

1.4.- Justificación

En años recientes realizar trabajos de investigación y desarrollo enfocados a sistemas avanzados de asistencia a la conducción, así como de movilidad autónoma se ha vuelto una misión social y ética; esto con el fin de ofrecer condiciones de manejo más seguras, además de prevenir accidentes de tránsito. De acuerdo con cifras emitidas por la Organización Mundial de la Salud, (OMS,2022) en un año mueren alrededor de 1.3 millones de personas debido a traumatismos causados por percances de tránsito, convirtiéndose en la principal causa de muerte de infantes y personas jóvenes de entre 5 y 29 años. Cabe mencionar que la cifra de personas que experimentan lesiones de carácter no mortal a causa de incidentes de tránsito supera el número de 20 millones de personas al año(OMS, 2022). En la mayoría de las víctimas las lesiones les causan algún tipo de discapacidad. Aunado a esto los daños por accidentes para la mayoría de los países tienen costos cercanos a el 3% de su PIB (OMS, 2022).

Según datos reportados por el Instituto Nacional de Estadística y Geografía INEGI (2022), en 2021 en México se presentaron 4,401 muertes debido percances de tráfico además de un total de 82,466 heridos a nivel nacional. Querétaro se encuentra entre las entidades federativas que en 2021 reportaron un rango de entre 126 a 206 muertes por accidentes de tránsito, además de un saldo de entre 288 a 1903 heridos. Por su parte el Instituto Nacional de Salud Pública (INSP) a través de un estudio realizado por investigadores de esta dependencia gubernamental analizaron la prevalencia a superar el límite de velocidad y los factores asociados a esta. En el estudio se evaluaron cuatro ciudades, Guadalajara-Zapopan (Jalisco), León (Guanajuato), Cuernavaca (Morelos) y Villahermosa (Tabasco), los resultados reportan que la prevalencia promedio al exceso de velocidad es del 47%, una cifra preocupante considerando que la probabilidad de sufrir traumatismos de gravedad es de más del 80% cuando la velocidad supera los 50 km/h. Por lo anterior el INSP advierte que si este comportamiento es similar en el resto de las ciudades del país una gran proporción de la población que transita por carreteras mexicanas se

encuentra expuesta a sufrir consecuencias graves a causa del exceso de velocidad, INSP (2020).

Ante tal situación de riesgo existe la necesidad de desarrollar un sistema que detecte y clasifique señales de tráfico mexicanas, el cual sea capaz, en una primera etapa de desarrollo, detectar solamente señalamientos viales preventivos en específico los que advierten zonas de riesgo como lo pueden ser curvas sinuosas, zonas peatonales, zonas con riesgo de derrumbe, etc. La elección de este tipo de señalamientos se da porque son los que con mayor frecuencia son ignorados por los conductores además de que son los que comúnmente se encuentran con más frecuencia en carreteras mexicanas.

Desde el punto de ingeniería el modelo propuesto será basado en el uso de redes neuronales de convolución. Esto debido a que este modelo tiene un tiempo de ejecución menor, además de que puede discriminar una cantidad mayor de información sin mencionar que puede clasificar los datos con mayor precisión que otros modelos existentes, de Jesús (2020).

El algoritmo de reconocimiento será programado en Python, este entorno de programación fue elegido ya que es de código abierto por lo que desde la parte económica representa una reducción de costos en el desarrollo de este proyecto. Además, este lenguaje de programación maneja librerías como OpenCV la cual sirve para el procesamiento de imágenes. TensorFlow y Keras son librerías que también están dentro del entorno de Python las cuales sirven para programar y entrenar modelos de redes neuronales convolucionales. La sintaxis de este lenguaje es muy fácil de aprender y manejar, esto fue un motivo más para elegir trabajar en este entorno de programación.

1.5.- Objetivos

1.5.1.- Objetivo General

Desarrollar un sistema de reconocimiento de señales de tránsito preventivas mexicanas programado en Python mediante aprendizaje de máquinas a través de redes neuronales convolucionales para la asistencia a la conducción.

1.5.2.- Objetivos particulares

a) Crear una base de datos a partir de fotografías que contengan cinco señales de tránsito preventivas tomadas en vialidades de San Juan del Río Querétaro, así como imágenes extraídas del buscador Google para el entrenamiento, prueba y evaluación del sistema.

b) Desarrollar un algoritmo de reconocimiento mediante el uso de técnicas de segmentación, detección y clasificación de imágenes capaz de distinguir las diferentes señales viales preventivas.

c) Programar en Python el algoritmo de reconocimiento a través de la librería OpenCV para entrenar, probar y evaluar el algoritmo.

d) Implementar el programa del algoritmo de reconocimiento en un sistema compuesto por una webcam y una computadora para detectar y reconocer señales viales preventivas en línea.

e) Desplegar los iconos de los señalamientos de tránsito restrictivos detectados por el sistema en una interfaz gráfica para que el sistema sea capaz de ofrecer mensajes visuales de asistencia a la conducción.

1.6.- Planteamiento general

La figura 1 muestra el diagrama de bloques del planteamiento general del proyecto, el cual está dividido en dos partes: los recuadros de color azul engloban las etapas de documentación, redacción, validación y análisis de resultados del proyecto mientras que los recuadros en color naranja se centran en el desarrollo del

algoritmo de detección y clasificación de señales de tránsito preventivas. A continuación, se describen brevemente las actividades a realizar.

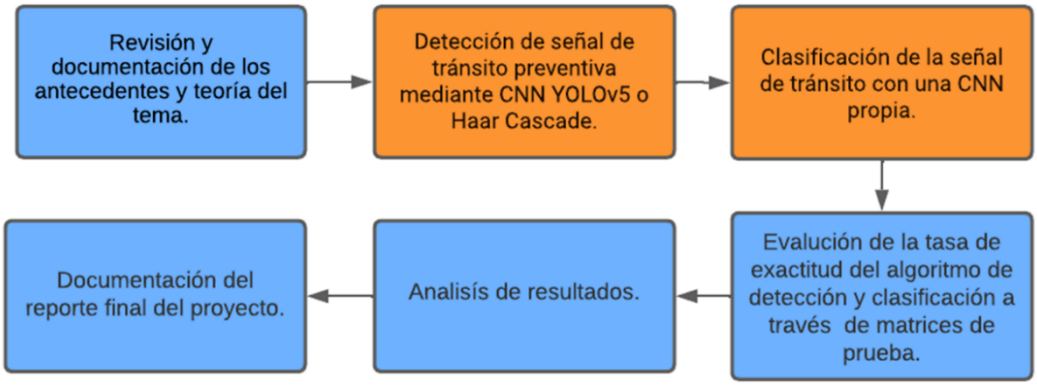

Figura 1.- Planteamiento general.

1.6.1.- Revisión y documentación de los antecedentes y la teoría del tema.

En esta etapa se hace la revisión de la documentación existente acerca de la implementación de redes neuronales de convolución para la detección y clasificación de señales de tránsito, se toman en cuenta documentos como artículos de congreso, artículos de revistas de divulgación científica y trabajos de tesis para la obtención de grado**.**

Para la fundamentación teórica se toman en cuenta distintas fuentes de información principalmente libros y páginas web oficiales que contengan los fundamentos teóricos de inteligencia artificial, visión artificial, redes neuronales biológicas y artificiales, así como la fundamentación matemática para el procesamiento de imágenes.

1.6.2.- Detección de señal de tránsito preventiva mediante CNN (YOLOv5x) y modelo Haar Cascade.

Esta sección del planteamiento general se realiza el desarrollo de un modelo que detecte señales de tránsito preventivas mediante el uso de una red neuronal de convolución ya existente y con un preentrenamiento para la detección

de objetos (YOLOv5x), esta red será entrenada con un set de datos de 1805 imágenes de distintos señalamientos viales preventivos. Adicionalmente se utilizará otro de modelo de detección utilizando la técnica de detección de objetos conocida como Haar Cascade.

1.6.3.- Clasificación de la señal de tránsito con una CNN propia.

Para esta etapa se llevará a cabo el desarrollo de una red neuronal de convolución propia, es decir que se programará desde cero una CNN utilizando lenguaje PYTHON y las librerías de Tensor Flow y Keras, con el objetivo de obtener un modelo que clasifique señales de tránsito preventivas en 5 clases diferentes.

1.6.4.- Evaluación de la tasa de exactitud del algoritmo de detección y clasificación a través de matrices de prueba.

En esta etapa de desarrollo se evalúa el modelo mediante la aplicación de la matriz de pruebas de la figura 2 en cada una de las señales de tránsito seleccionas para el estudio.

Detección de señales preventivas de tránsito					
Número de pruebas	Condiciones de iluminación	Condiciones climáticas	Nivel de desgaste en la señal de tránsito	Velocidad del vehículo	Horario de prueba
10	Completamente iluminado	Día soleado	Señal en condiciones nominales	40 km/h	12:00 a 14:00
10	Completamente iluminado	Día soleado	Señal en condiciones nominales	60 km/h	12:00 a 14:00
10	Completamente iluminado	Día soleado	Señal en condiciones nominales	80 km/h	12:00 a 14:00
10	Completamente iluminado	Día soleado	Señal en condiciones nominales	100 km/h	12:00 a 14:00

Figura 2.- Matriz de pruebas.

1.6.5.- Análisis de resultados.

Una vez que se apliquen todas las matrices de prueba se llevará a cabo un estudio estadístico de las pruebas para obtener la tasa de exactitud del modelo de detección y clasificación de señales de tráfico preventivas bajo las condiciones propuestas para el estudio.

1.6.6.- Documentación del reporte final del proyecto.

Como última etapa de este proyecto se llevará a cabo la redacción del reporte final o documento de tesis, en donde se reporten los resultados y conclusiones derivadas del presente trabajo de investigación y desarrollo.

Capítulo 2. Fundamentación teórica y tecnológica

2.1.- Inteligencia artificial

Para poder definir a la inteligencia artificial (IA) es necesario primero tener un concepto claro de lo que es la inteligencia. Hoy en día todavía no existe un consenso entre ingenieros y científicos acerca de qué describe exactamente a la inteligencia artificial debido a que no se tiene la seguridad o certeza de qué es lo que hace o no inteligente a un ente (García, 2012). Según la Real Academia Española (RAE) la inteligencia se define como aquella capacidad para entender y comprender, así como también la capacidad para poder resolver problemáticas (RAE, 2022). La inteligencia humana presenta ciertas propiedades generales a destacar, como por ejemplo la capacidad de elaborar planes, de resolver problemas, de enfrentar nuevas situaciones y responder preguntas. Tomando en cuenta lo anterior, la IA es esa búsqueda para poder modelar los comportamientos del pensamiento humano y aplicar dichos modelos en sistemas que sean capaces de resolver problemas como lo haría una persona (Ponce, 2010).

Para que un sistema de IA pueda resolver un problema es necesario que cuente con una secuencia finita de pasos o instrucciones que de manera detallada dicte las acciones que el computador debe realizar o ejecutar para dar una solución a dicho problema. Todo este conjunto de instrucciones forma lo que se conoce como estructura algorítmica del sistema de IA (Benítez & Escudero, 2013).

Para fines prácticos de este proyecto la IA se define como el conjunto de técnicas, algoritmos y herramientas computacionales que sirven para crear sistemas que toman decisiones basadas en el comportamiento y pensamiento humano y que están destinados para resolver problemas que por defecto requieren el uso de cierto grado de inteligencia (García, 2012).

2.2.- Aprendizaje automático

El aprendizaje automático o aprendizaje de máquinas (del inglés machine learning) es el campo de la IA que busca darle a un computador las herramientas necesarias para que este pueda aprender por sí mismo a darle solución a un problema. De acuerdo con la definición dada por Arthur Samuel de IBM, el

aprendizaje automático es la disciplina que dota de capacidades a una computadora para esta sea capaz de dar solución a problemáticas para las cuales no ha sido programada explícitamente (Berzal, 2018).

En general el machine learning es la rama de la IA que tiene como objetivo que un ordenador aprenda a través de una base de datos de entrada, los cuales sirven de entrenamiento para encontrar patrones dentro de la base de datos. Una vez que el modelo de machine learning termina el entrenamiento estos patrones son utilizados para realizar predicciones con nuevos datos nunca vistos (Pineda, 2021).

2.3.- Aprendizaje supervisado

Dentro del aprendizaje automático existe lo que se conoce como aprendizaje supervisado, la filosofía que utiliza el modelo para adquirir conocimiento se basa en el aprendizaje a partir de ejemplos, es decir que los ejemplos utilizados en el entrenamiento son marcados con lo que se conoce como etiqueta, esta etiqueta contiene la información que caracteriza los datos. El modelo de aprendizaje supervisado aprende asociando los datos de entrenamiento con su respectiva etiqueta (Berzal, 2018).

2.4.- Redes neuronales biológicas

Básicamente una neurona biológica es un tipo de célula que se especializa en procesar y transmitir información. La estructura de una neurona biológica está compuesta por lo que se conoce como “soma” el cual es el cuerpo de la célula, además de dos ramificaciones conocidas como axón y dendritas. El proceso que sigue una neurona para recibir y transmitir información consiste primeramente en la recepción de las señales por parte de otras neuronas en forma de impulsos de origen químico a través de las dendritas. Posteriormente, para transmitir la información que produce la neurona, el soma o cuerpo de la célula envía impulsos eléctricos a lo largo del axón. La estructura de una neurona biológica es la que se muestra en la figura 3 (Ponce, 2010).

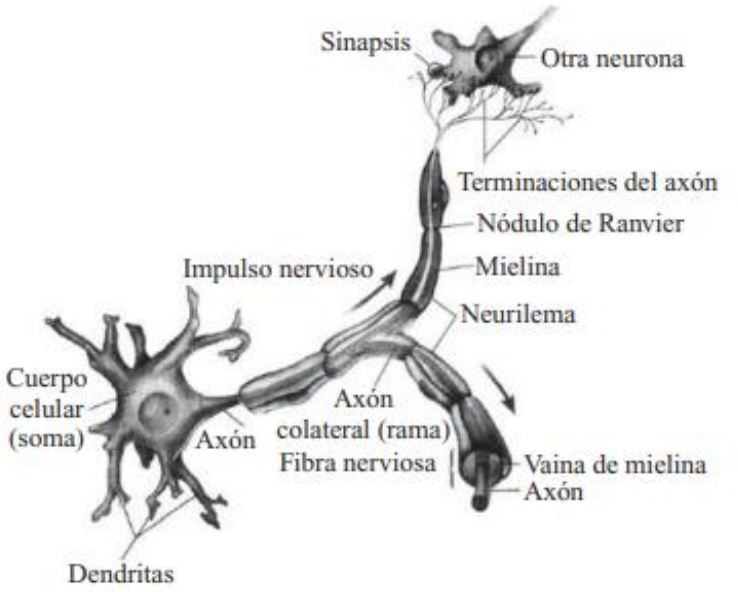

Figura 3.- Estructura de una neurona biológica (Ponce, 2010).

El parentesco directo que tienen las redes neuronales biológicas y artificiales se basa en la actividad sináptica y la analogía de las redes neuronales artificiales, en el cual las señales o impulsos que llegan a la sinapsis son las entradas de la neurona; estas entradas se ponderan de acuerdo a un parámetro denominado peso, donde cada entrada tiene asociado un peso diferente y que dependiendo del signo positivo o negativo de la entrada activa o inhibe la excitación de la neurona, por lo que la salida de la neurona es la suma ponderada de sus entradas, si esta suma sobrepasa el valor de umbral se activa la neurona (Ponce, 2010).

2.5.- Redes neuronales artificiales

Una red neuronal artificial (RNA) es una aproximación de la operación del cerebro humano, por lo que no es apropiado pensar que los principios de funcionamiento de las redes artificiales son iguales al de las biológicas, ya que estas solo intentan emular una parte bastante simple del comportamiento del pensamiento humano (Ponce, 2010).

El modelo en el que se basa una red neuronal artificial se inspira en la forma en la que las redes neuronales biológicas procesan la información que recibe el cerebro. La estructura de un RNA es el elemento clave del modelo, donde esta estructura está compuesta por los elementos de procesamiento conocidos como neuronas que trabajan en conjunto para resolver un problema en específico (Palma & Marín, 2008).

El modelo matemático que siguen actualmente las neuronas artificiales es el planteado por McCulloch y Pitts en 1943. Este modelo matemático básicamente plantea que cada neurona recibe una colección de entradas $\{x_1, x_2, \dots, x_D\}$ y regresa una salida única y. Dependiendo del modelo una RNA puede contener enormes cantidades de conexiones entre sus neuronas las cuales de cierto modo simulan las interconexiones neuronales de un cerebro. Al igual que una red neuronal biológica una RNA puede establecer en menor o mayor medida la intensidad de activación de cada una de las neuronas, esta intensidad de activación está determinada por los pesos que están asociados a cada entrada, por lo que cada entrada x_i se ve afectada por un peso w_i, ver figura 4 (Palma & Marín, 2008).

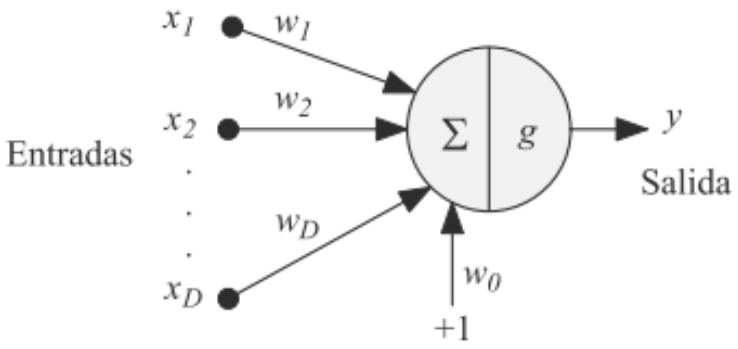

Figura 4.- Modelo matemático de una neurona de McCulloch-Pitts (Palma & Marín, 2008).

La expresión matemática para obtener la salida y de una neurona se hace a través de la suma ponderada a de las entradas la cual se conoce como la activación de la neurona:

$$a = \sum_{i=1}^{D} w_i x_i + w_o \quad (1)$$

Donde w_0 es el umbral o sesgo que es utilizado para equilibrar el valor promedio de las entradas, sobre todo en el grupo de entrenamiento y el correspondiente valor promedio de las salidas deseadas. Una vez que se obtiene el valor de a, la salida y se obtiene al aplicar la función conocida como función de activación o de transferencia $g(a)$, como se muestra a continuación:

$$y = g(a) = g\left(\sum_{i=1}^{D} w_i x_i + w_o\right) = g\left(\sum_{i=0}^{D} w_i x_i\right) \quad (2)$$

Donde el umbral w_0 se puede tomar como un peso adicional si se supone una entrada extra x_0 que tenga un valor fijo de 1 (ver figura 4). Para finalizar, la ecuación (2) se puede reescribir de manera vectorial como:

$$g(a) = g(w^T x) \quad (3)$$

Donde w es el vector de pesos y x es el vector de las entradas a la red (Palma & Marín, 2008).

2.6.- Funciones de activación

En los modelos de redes neuronales artificiales existe la combinación de sus entradas con los pesos que tienen asociados. Para obtener esta combinación de entradas y pesos es necesario realizar la suma ponderada de las entradas multiplicadas por su peso asociado. Esto da como resultado lo que se conoce como

entrada neta o excitación de la neurona. A esta entrada neta también se le suma el sesgo de la neurona para compensar el valor medio de las entradas (Berzal, 2018).

Normalmente la salida de la neurona no está dada por la entrada neta y la suma del sesgo. Generalmente se le aplica algún tipo de transformación no lineal a estos parámetros. Si se construyen redes neuronales de múltiples capas que tengan como salida una función lineal, el resultado final de la combinación de todas las capas daría como resultado una función lineal también. Esto se traduciría en tener solamente una capa de neuronas lineales. En la mayoría de los problemas reales que existen se encuentran no linealidades asociadas a este tipo de problemas. Por lo que es de gran interés que las redes neuronales sean capaces de procesar y representar dichas no linealidades (Berzal, 2018).

Una vez que se obtiene la entrada neta y la suma de sesgo se le aplica lo que se conoce como función de activación o transferencia, esta función de activación establece el valor de la salida de la neurona, buscando siempre eliminar linealidades en la salida que produzca la simplificación de múltiples capas de redes neuronales lineales a una única capa de neuronas lineales (Berzal, 2018).

Las funciones de activación más utilizadas son las que se observan en la figura 5.

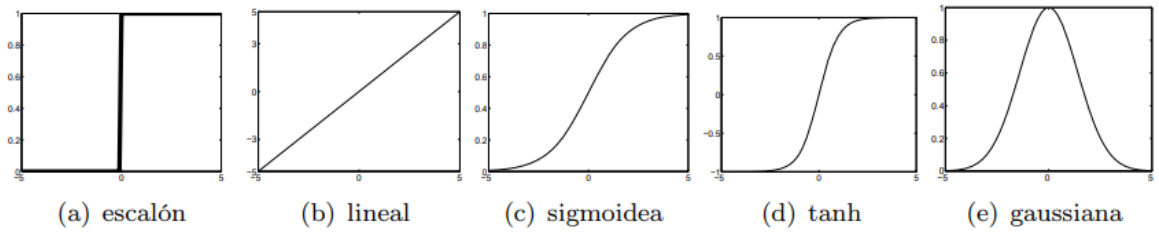

Figura 5.- Funciones de activación (Palma & Marín, 2008).

Estas funciones de activación están representadas por las siguientes expresiones:

- Escalón

$$g(a) = \{\, 0 \qquad a < 0\; 1 \qquad a > 0 \tag{4}$$

- Lineal

$$g(a) = a \tag{5}$$

- Sigmoidea

$$g(a) = \frac{1}{1+e^{-a}} \tag{6}$$

- Tangente hiperbólica

$$g(a) = tanh = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}} \tag{7}$$

- Gaussiana

$$g(a) = exp\left(-\frac{(a-\mu)^2}{2\sigma^2}\right) \tag{8}$$

Actualmente las funciones de activación que más se utilizan en redes neuronales de convolución son las funciones: ReLU y Softmax.

ReLU es una función que modifica los valores de entrada. De modo que anula aquellos valores negativos y deja los valores positivos tal y como ingresan, (Calvo, 2018). Está función está representada por la siguiente expresión:

$$f(x) = (0, x)\; = \{0 \qquad x < 0\; x \qquad x \geq 0 \tag{9}$$

La función Softmax por su parte modifica las salidas a una representación en forma de probabilidades, de modo que se obtiene un valor de 1 en la suma de todas las probabilidades de las salidas (Calvo, 2018). Se representa mediante la siguiente expresión:

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \tag{10}$$

2.7.- Convolución de matrices

La convolución de matrices es la base de algunos procedimientos para el filtrado de imágenes. Al aplicar una convolución en una imagen, el valor de un píxel de salida está dado por la suma ponderada de sus pixeles vecinos. Este proceso de convolución en una imagen se realiza entre la matriz de los pixeles de la imagen y una matriz que es un filtro conocido como kernel (Esqueda, 2002).

Al aplicar un proceso de convolución a una matriz utilizando un kernel de 3x3 (ver figura 6) se observa que recoge los resultados de las correspondientes convoluciones en las entradas (2,2) y (3x2) respectivamente. En la figura 7 se aprecia el resultado final de la convolución donde los valores de los alrededores, es de decir de la primer y última fila, así como de la primera y última columna de la matriz se mantienen iguales, aunque existen otras opciones para completar los valores de estas casillas como puede ser: completar con ceros los valores de alrededor o en su defecto completar los valores de la parte simétrica opuesta, (Giménez et al.,2016).

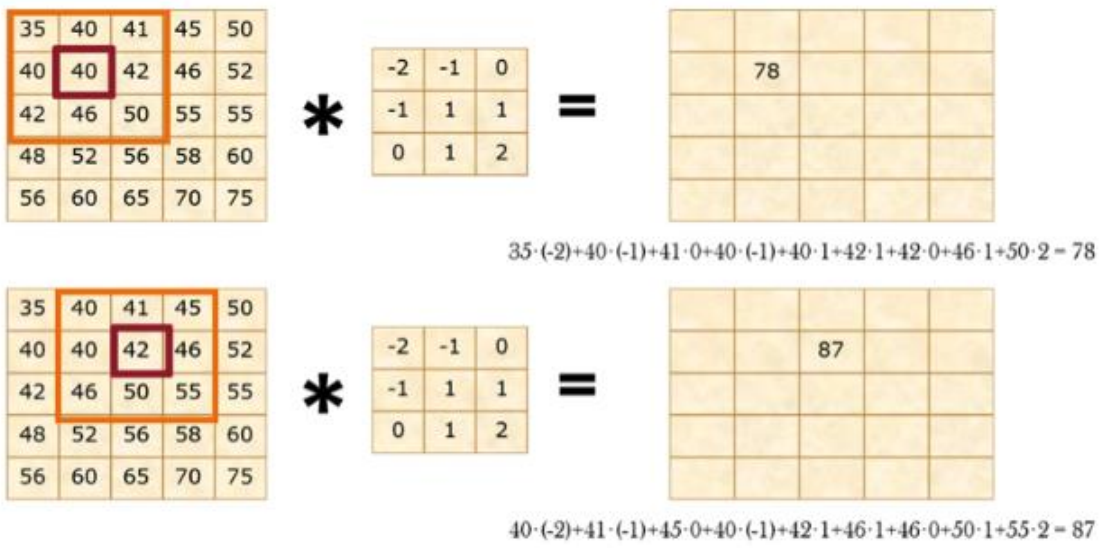

Figura 6.- Proceso de convolución (Giménez et al.,2016).

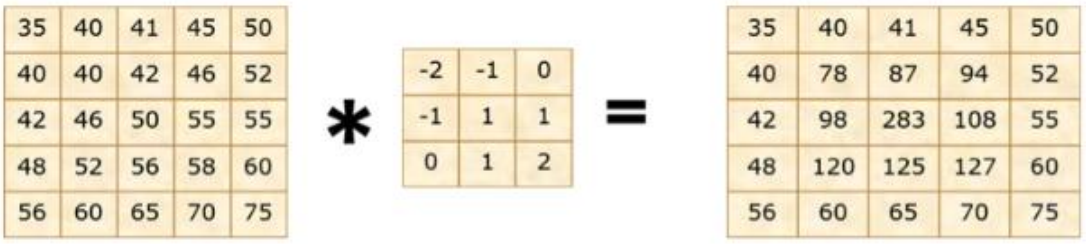

Figura 7.- Resultado de aplicar la convolución a una matriz (Giménez et al.,2016).

En la figura 8 se observan los filtros kernel que más se utilizan en el procesamiento de imágenes (Giménez et al.,2016).

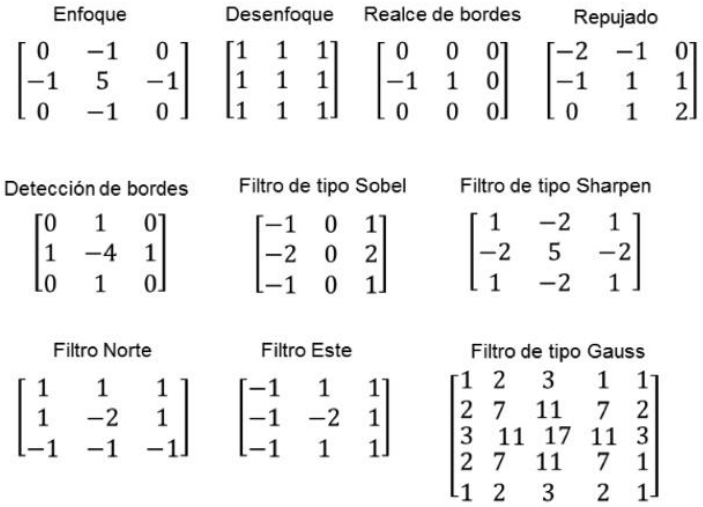

Figura 8.- Filtros kernel más utilizados en el campo de procesamiento de imágenes (Giménez et al.,2016).

2.8.- Redes neuronales de convolución

Las redes neuronales convolucionales CNN (del inglés Convolutional Neural Network) es un modelo de red neuronal que de cierto modo comparte parentesco con el perceptrón multicapa. La principal diferencia radica en el uso de las capas de convolución y del método de submuestreo (del inglés subsampling) el

cual es utilizado para reducir el tamaño de los datos seleccionando un subconjunto de los datos originales (García, 2019).

Este tipo de redes neuronales son desarrolladas específicamente para el procesamiento de datos bidimensionales donde su principal aplicación es el reconocimiento de imágenes, aunque también puede reconocer señales de voz en espectrogramas (García, 2019).

La estructura de una CNN se basa en una arquitectura que consiste en la concatenación de una capa de convolución seguida por una capa de reducción o del inglés pooling (ver figura 9). Esta capa de pooling utiliza el método de submuestreo o subsampling para la reducción de datos, este proceso se vuelve iterativo hasta que se obtiene un set o conjunto de matrices, de tal manera qué al colocar todos los elementos dentro de un vector tenga una cantidad considerablemente viable de elementos para que computacionalmente se pueda introducir como una entrada a una red neuronal (García, 2019).

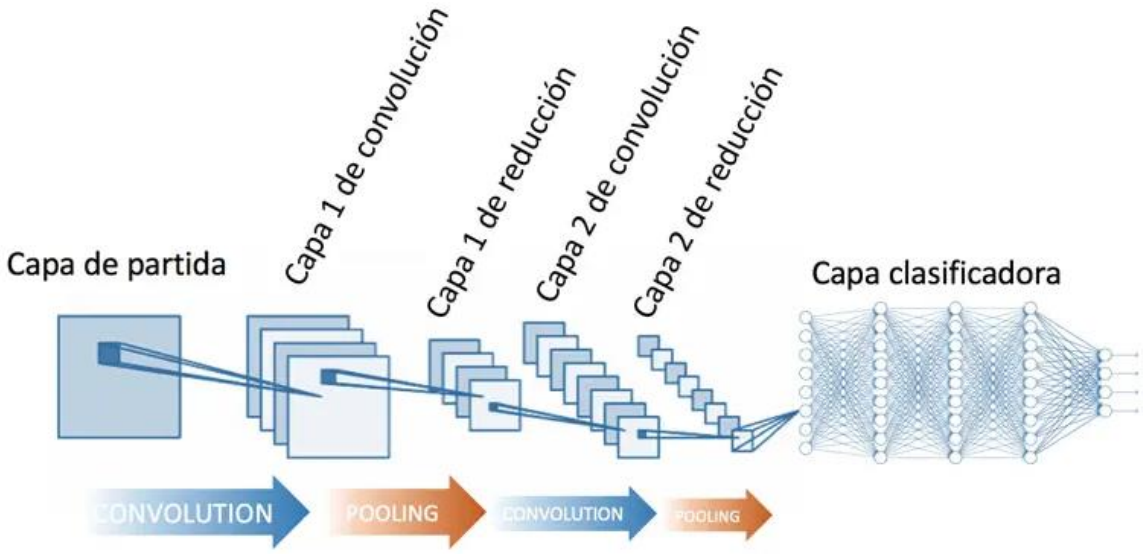

Figura 9.- Estructura de una CNN (Calvo, 2022).

2.9.- Visión artificial

Se define como visión artificial a la disciplina de la IA que conjunta las distintas técnicas que permiten el procesamiento de imágenes a partir de su captura y análisis, con el objetivo de extraer las características más importantes de la imagen. La visión artificial busca reemplazar mediante la automatización ciertos procesos o acciones que hasta hace poco estaban destinadas solamente a la actividad humana, como lo son tareas de seguridad, industria, comercio, medicina, etc. (Domínguez, 2021).

La visión artificial consiste principalmente en deducir de manera automática la estructura y características de un espacio en tercera dimensión, a partir de una o varias imágenes bidimensionales de ese espacio, cabe mencionar que es posible que ese espacio o mundo en 3D sea dinámico también (Sánchez & Ricolfe, 2016).

2.10.- Norma Oficial Mexicana NOM-034-SCT2-2011 "Señalamiento horizontal y vertical de carreteras y vialidades urbanas"

2.10.1.- Señalamiento

La Secretaria de Comunicaciones y Transportes a través de la Dirección General de Conservación de Carreteras (SCT b,2022) y basándose en lo que indica la Norma Oficial Mexicana NOM-034-SCT2-2011 menciona que un señalamiento es el conjunto de señales que indiquen la geometría de las carreteras y vías de la ciudad y sus divisiones en el camino, intersecciones y pasos a nivel; advierten sobre peligros potenciales en el camino y su naturaleza; regula el tráfico, indica restricciones físicas y legales que limitan el uso estos caminos públicos; denota elementos estructurales instalados en la vía; y guía el viaje del usuario, (SCT c, 2022).

La NOM-034-SCT2-2011 menciona que existen dos tipos de señalamientos:

- Señalamientos horizontales.
- Señalamientos verticales.

2.10.2.- Señalamiento horizontal

Los señalamientos horizontales (ver figura 10) son el un conjunto de marcas pintadas o colocadas sobre pavimentos, acabados y armaduras con la finalidad de identificar las propiedades geométricas del asfalto o calles de la ciudad; además de resaltar todos los elementos de construcción que se instalan a la derecha de las vialidades los cuales sirven para la regulación y control del tráfico de vehículos y peatones, así como proporcionar información a los usuarios. Estos signos son rayas, símbolos, leyendas o dispositivos (SCT c, 2022).

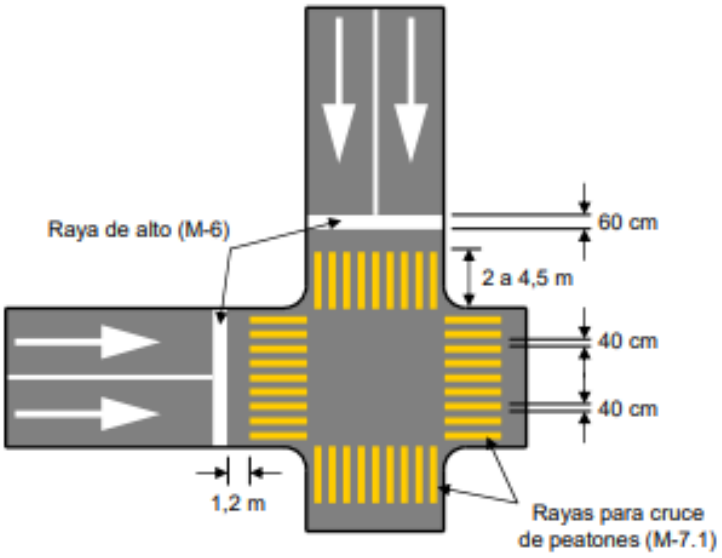

Figura 10.- Ejemplo de señalamiento horizontal de tránsito (SCT c, 2022).

2.10.3.- Señalamiento vertical

Es la colección de señales en tableros fijados en postes, marcos y otras estructuras, integradas con leyendas y símbolos. De acuerdo con su objetivo, las señales que existen en carreteras mexicanas pueden ser:

- Preventivas.
- Restrictivas.
- Informativas.
- Turísticas y de servicios.
- Diversas.

2.10.4.- Señalamiento preventivo

Son señalamientos que tienen como propósito prevenir al conductor ante las posibles zonas o puntos de riesgo potencial en el camino y su naturaleza (ver figura 11), (SCT c, 2022).

Figura 11.- Ejemplo de señalamiento preventivo (SCT a, 2022).

2.11.- Python

Python (ver figura 12) es básicamente un lenguaje de programación de código abierto e interpretado, esto se refiere a que un intérprete va leyendo cada una de las líneas de código que se escriben a su vez que se van ejecutando. Esto ahorra tiempo de compilación debido a que los programas no tienen que compilarse previamente lo cual dota de más rapidez de ejecución a los programas. Una ventaja de Python es que es de multiplataforma de manera que un programa puede compilarse en distintas plataformas o sistemas operativos sin mayor problema. Otra ventaja de este lenguaje de programación es su sintaxis, la cual es bastante sencilla y fácil de aprender, además de contar con un gran número de librerías que constantemente son actualizadas. Es por estas propiedades que Python es el lenguaje de programación predilecto de los desarrolladores de programas de aprendizaje automático (Pineda, 2021).

Figura 12.- Logo Python (Python, 2022).

2.12.- OpenCV

OpenCV es una librería de Python que a su vez es una interfaz de programación de aplicaciones API (del inglés *Application Programming Interface*) que contiene aproximadamente 300 funciones escritas en lenguaje C (Arévalo et al. 2002).

Las características que presenta esta librería son las siguientes:

- Es de uso libre por lo que se puede utilizar tanto comercialmente como no comercial.
- No requiere el uso librerías numéricas externas, a pesar de que cuenta con la capacidad de utilizar algunas de ellas si es que se encuentran disponibles en el momento de ejecución.
- Cuenta con interfaces para lenguajes y entornos diferentes.

Esta librería esta principalmente dirigida a la visión artificial en tiempo real. Entre sus aplicaciones se encuentran: identificación de objetos, segmentación, reconocimiento de gestos, seguimiento de movimiento, estructura del movimiento y robots móviles (Arévalo, 2002).

2.13.- PyCharm

PyCharm (ver figura 13) es un Entorno de Desarrollo Integrado IDE (del inglés *Integrated Development Environment*) para el desarrollo de programas que utilizan código Python. Esta herramienta de software provee de un amplio repertorio de instrumentos indispensables para los programadores que eligen Python como su lenguaje de programación (Jet Brains, 2022).

PyCharm cuenta con un asistente inteligente que realiza inspecciones del código, indicaciones de errores sobre la marcha y correcciones rápidas, además de refactorizar el código de manera automática, así como completar funciones de navegación (Jet Brains, 2022).

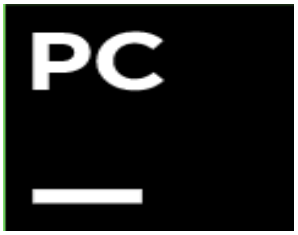

Figura 13.- Icono de PyCharm (Jet Brains, 2022).

2.14.- You Only Look Once (YOLO)

YOLO es una arquitectura de software que tiene como objetivo detectar objetos. El principal responsable de su creación es Joseph Redmond. Esta arquitectura es bastante rápida y precisa lo cual la hace óptima para la detección de objetos en tiempo real mediante la captura de video (Rozada, 2021).

La base de esta arquitectura utiliza una red neuronal que trabaja por secciones sobre una imagen para extraer las características más importantes y trazar las cajas delimitadoras (del inglés *bounding boxes*), esto crea una probabilidad de detección por categoría para cada bounding box (Treviño, 2022). Con esto se consigue una red neuronal que responde ante los objetos presentes en una imagen (Rozada, 2021).

2.14.1.- YOLOv5

YOLOv5 es la colección de modelos y arquitecturas de software que se utilizan para detectar de objetos. Estos modelos son entrenados previamente haciendo uso de la base de datos conocida como “Objetos Comunes en Contexto” COCO (del *inglés Common Objects In Context*) y representan los métodos de investigación de código abierto de Ultralytics sobre enfoques hacia IA con visión de futuro, anexando miles de horas de lecciones aprendidas y mejores prácticas. La estructura de la red YOLOv5 se muestra en la figura 14 (Jocher, 2020).

YOLOv5 consta de cuatro versiones propias, que son YOLOv5s, YOLOv5m, YOLOv5l y YOLOv5x. Se clasifican según el tamaño de la memoria, pero el principio de funcionamiento es el mismo. YOLOv5x tiene el mayor tamaño de almacenamiento, y YOLOv5s tiene el menor tamaño de almacenamiento (Jung & Choi, 2022).

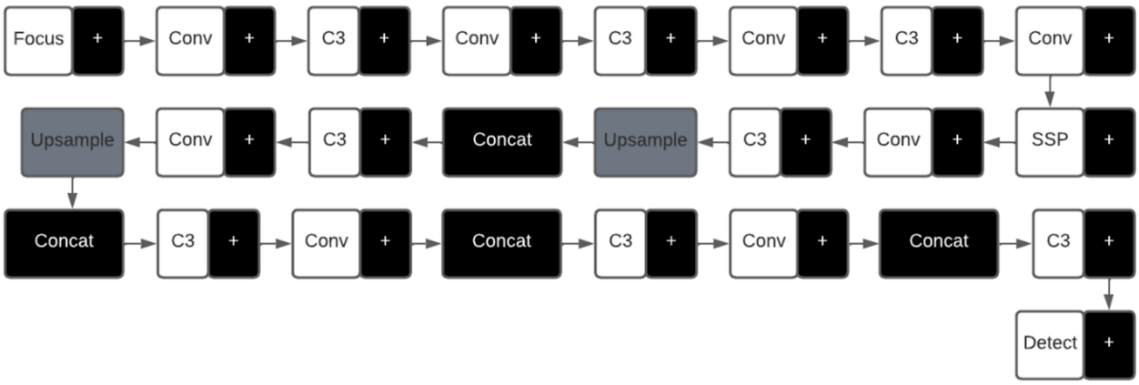

Figura 14.- Estructura de red YOLOv5 adaptado de Gutta (2021).

2.15.- Haar Cascade

Haar Cascade es una técnica de aprendizaje de máquinas que permite realizar la detección o reconocimiento de objetos en una imagen o en los fotogramas de un video. Esta técnica fue propuesta por Viola y Jones en el año de 2001 (Jeremías, 2020). El proceso que sigue un clasificador es el que se muestra de forma esquemática en la figura 15. En la figura 15 los círculos marcados con los números 1, 2 y 3 representan sub-ventanas, a cada una de estas sub-ventanas se le aplica una serie de clasificadores. La primer sub-ventana discrimina un gran número de entradas negativas es decir imágenes que no contienen el objeto que se desea detectar utilizando muy poco procesamiento. Si la primer sub-ventana determina que la entrada es positiva (que el objeto de interés se encuentra dentro de la imagen de entrada) pasa a la segunda sub-ventana, cada capa o sub-ventana posterior utiliza mayor capacidad de procesamiento que la anterior para determinar si el dato es positivo o negativo (Viola & Jones, 2001). Estos clasificadores se denominan débiles debido a que la probabilidad de que arrojen falsos positivos es muy alta. Sin embargo, la concatenación por sub-ventanas de estos clasificadores da como resultado un sistema robusto de detección de objetos (Jeremías, 2020).

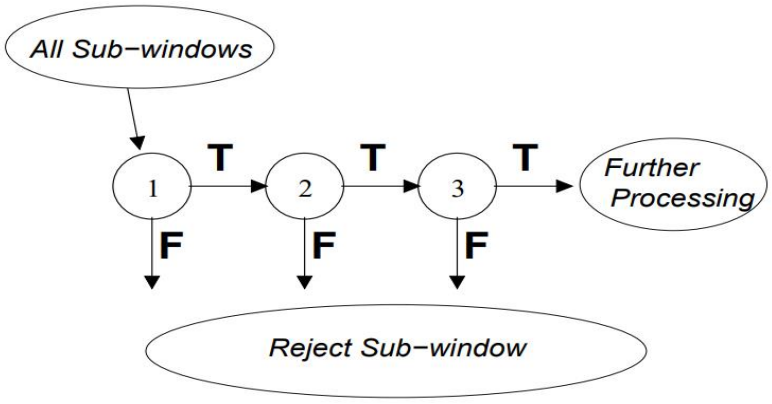

Figura 15.- Representación esquemática de una cascada de detección (Viola & Jones ,2001).

El algoritmo propuesto por Viola y Jones utiliza ventanas del mismo tamaño (ver figura 16) las cuales pueden tener 2, 3 o 4 rectangulos de igual tamaño. Lo que hace el algoritmo es aplicar la funcion Haar en cada ventana, esta funcion calcula la suma de los pixeles que se encuentran dentro de los rectangulos blancos y le resta la suma de los pixeles dentro de los rectangulos sombreados. De esta manera el algoritmo determina las caracterisiticas del objeto de interes dentro de la imagen comparando las imágenes positivas de las negativas (Jeremías, 2020).

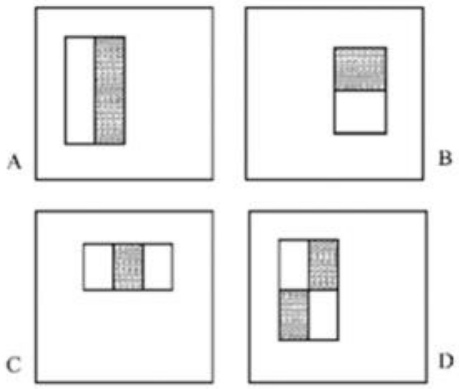

Figura 16.- Representación esquemática de una cascada de detección (Viola & Jones ,2001).

2.16.- Norma ISO/IEC 25010

La norma ISO/IEC 25010 es una norma perteneciente a la familia de normas ISO/IEC 25000. Esta familia de normas tiene por objetivo desarrollar un marco común de trabajo para la evaluación de la calidad en producto de software. La norma ISO/IEC 25010 describe la calidad de los productos de software y su uso. Este estándar internacional describe las características y sub características de calidad para evaluar productos de software. El modelo de calidad del producto definido por ISO/IEC 25010 consta de ocho características de calidad:

- Adecuación funcional.
- Eficiencia de desempeño.
- Compatibilidad.
- Usabilidad.
- Fiabilidad.
- Seguridad.
- Mantenibilidad.
- Portabilidad.

La calidad de un producto de software se puede interpretar como el grado en que el producto cumple con los requisitos de sus usuarios y, por lo tanto, proporciona valor. Son estos requisitos reflejados en el modelo de calidad (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que dividen la calidad del producto en características y sub características (ISO25000, 2023).

Capítulo 3. Metodología

3.1.- Descripción general de la metodología

En esta sección de la metodología del proyecto se describen detalladamente las actividades a seguir durante el desarrollo del modelo de detección y clasificación de señales preventivas de tráfico. En la Figura 17 se muestra el diagrama a bloques de la metodología del proyecto.

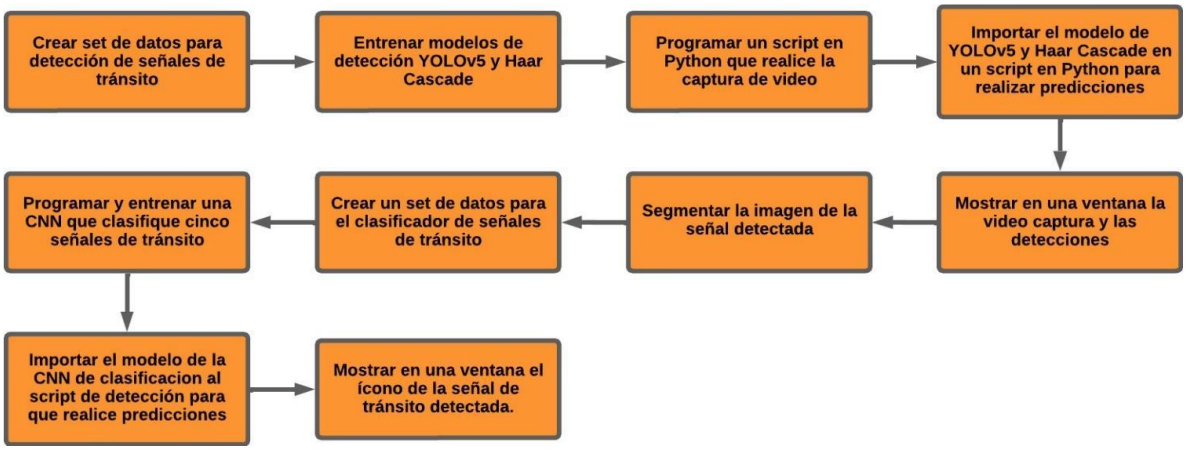

Figura 17.- Metodología del proyecto.

3.2.- Desarrollo de conjunto de datos de entrenamiento para los modelos de detección se señales de tránsito

En esta primera etapa del desarrollo del sistema de reconocimiento de señales mexicanas de tránsito preventivas es necesario la recolección de un conjunto datos que sirva para el entrenamiento y validación de los dos modelos de detección de señalamientos de tráfico. En este proyecto se utilizan dos modelos de detección de objetos: YOLOv5 y Haar Cascade.

3.3.- Conjunto de datos para entrenamiento de modelo YOLOv5

Para el modelo YOLOv5 se requiere un conjunto de datos que contenga imágenes de entornos donde se encuentre el objeto a detectar, en este caso particular se requieren imágenes con presencia de señalamientos mexicanos de tránsito. Para hacer la adquisición de las imágenes del set de datos (ver figura 18) se programa un script en Python que realice captura de video, y que sea capaz de extraer y guardar en una carpeta cada uno de los fotogramas de un video de carretera que contenga señalamientos mexicanos de tráfico.

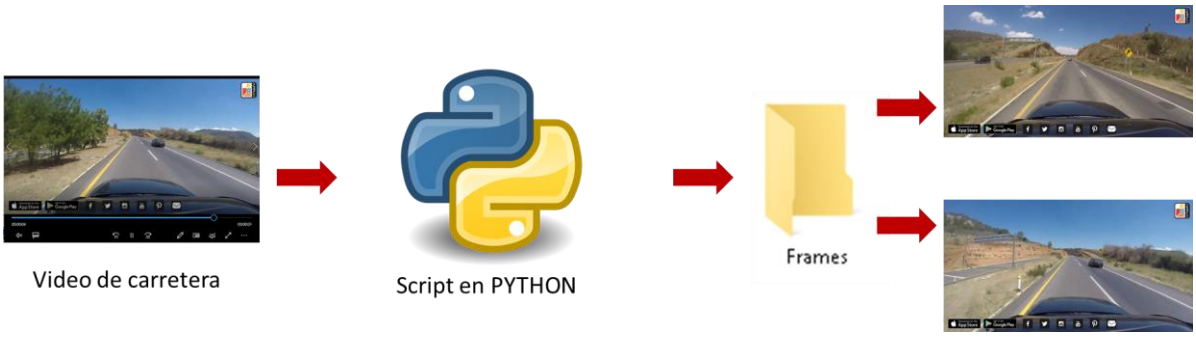

Figura 18.- Adquisición de imágenes para conjuntos de datos de entrenamiento del modelo YOLOv5.

Una vez que se adquieren todos los fotogramas del video de carretera es necesario revisar cada una de las imágenes almacenadas por el script y eliminar todas aquellas imágenes que no contengan señales de tránsito, este paso se hace de forma manual. El siguiente paso es dividir el conjunto de datos en dos carpetas, con los nombres: entrenamiento y validación, estas carpetas se almacenan en otra carpeta nombrada como “imágenes”. Del total de imágenes recolectadas el 80% se almacenan en la carpeta de entrenamiento y el 20% restante en la carpeta de validación.

3.4.- Etiquetado del conjunto de datos

Para llevar a cabo el etiquetado de los datos es necesario hacer uso de la herramienta web Make Sense, la cual es ampliamente utilizada en el etiquetado de conjuntos de datos de redes neuronales YOLO. Para etiquetar los datos es necesario cargar en la página web la carpeta que contenga las imágenes de interés. Para este caso particular se requiere etiquetar los datos almacenados en las carpetas de entrenamiento y validación por lo que el primer paso es cargar la carpeta de entrenamiento en la herramienta Make Sense. Una vez hecho lo anterior se debe elegir la opción de detección de objetos, además de asignar una etiqueta para los objetos a detectar por el modelo. Lo que resta es encerrar en un recuadro el objeto de interés en la imagen y asignarle la etiqueta. Este proceso es manual y se hace para cada una de las imágenes del conjunto de datos (ver figura 19).

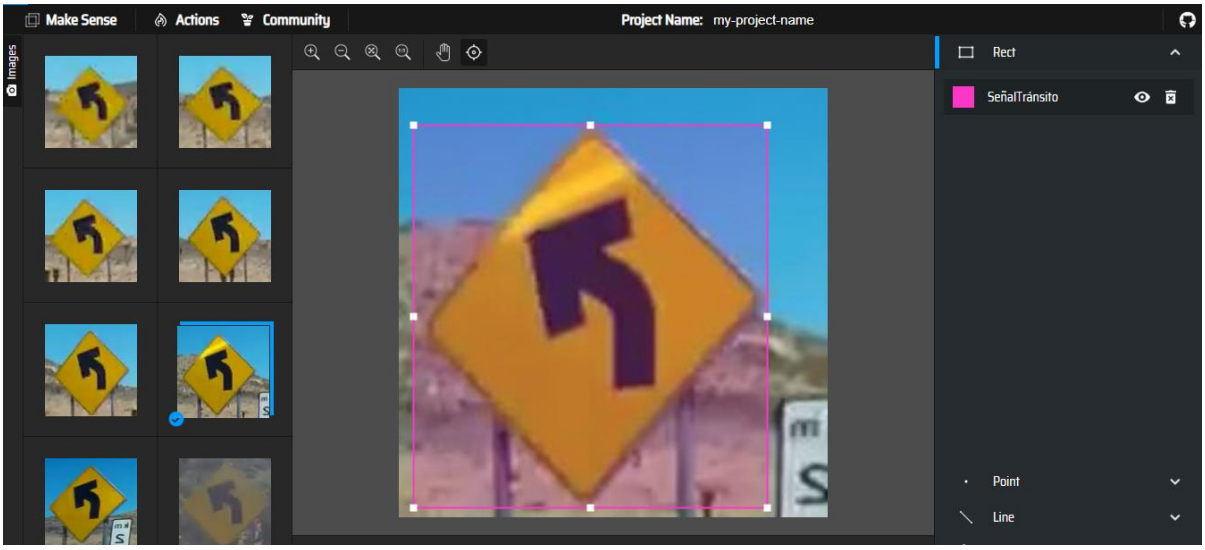

Figura 19.- Etiquetado de imágenes en Make Sense.

Después de que se han etiquetado todas las imágenes se necesita exportar las anotaciones hechas por Make Sense, al momento de exportar se debe elegir el formato de YOLO para los archivos devueltos por la herramienta web. Make Sense devuelve un archivo ZIP el cual contiene archivos txt con las etiquetas de cada una

de las imágenes de la carpeta de entrenamiento. Este proceso de etiquetado se repite para la carpeta de validación. Posteriormente se crea una carpeta con el nombre “Datos” en esta carpeta se guarda la carpeta “imágenes” la cual contiene las imágenes de entrenamiento y validación, a su vez se crea otra carpeta con el nombre “etiquetas” en esta carpeta se crean dos carpetas más: “entrenamiento” y “validación” en cada una de estas carpetas se guardan los archivos txt con las etiquetas de entrenamiento y validación.

3.5.- Conjunto de datos para entrenamiento de modelo Haar Cascade

Para el conjunto de datos del modelo Haar Cascade se requieren dos conjuntos de imágenes: un conjunto con imágenes positivas, es decir que contengan el objeto de interés en este caso señales de tráfico y un conjunto con imágenes negativas donde el objeto a detectar no este visible por ninguna parte de la imagen. Para guardar estos dos conjuntos de imágenes se crea una carpeta con el nombre de “DatasetHC” dentro de esta carpeta se crean dos más: “p” y “n”, en la carpeta “p” se guardan las imágenes positivas y en “n” las negativas. Para realizar la adquisición de las imágenes se utiliza el script de extracción de fotogramas que se utiliza para la adquisición de datos del modelo YOLOv5.

3.6.-Entrenamiento de modelos de detección YOLOv5 y Haar Cascade

En esta etapa se describe el proceso de entrenamiento de los modelos de detección de señales de tránsito YOLOv5 y Haar Cascade. Para el entrenamiento del modelo YOLOv5 se hace uso de la herramienta Google Colab; el entrenamiento del modelo Haar Cascade requiere el uso del software Cascade-Trainer- GUI.

3.6.1.- Entrenamiento YOLOv5

De acuerdo con la documentación oficial de la red YOLOv5, para el entrenamiento de un modelo de detección de objetos se requiere el uso de la herramienta de software Google Colab la cual es de código abierto por lo que no existen restricciones para su uso. Dentro de esta documentación existe un apartado para el entrenamiento de la red, este apartado proporciona un código ejecutable en Google Colab para entrenar la red YOLOv5, a este código se le tiene que importar

la carpeta “Datos” la cual contiene las carpetas con los conjuntos de datos de entrenamiento y validación, así como las carpetas con las etiquetas del conjunto de datos. Una vez realizado lo anterior es necesario configurar los parámetros de entrenamiento de la red como lo es el número de épocas de entrenamiento, este parámetro sirve para dictaminar el número de ciclos que el modelo leerá todos los datos de entrenamiento. Otro parámetro relacionado con las épocas es el tamaño del lote (del inglés batch size), el cual determina el número de ejemplos que se le muestran a la red en cada iteración de aprendizaje. Por último, es necesario elegir uno de los cinco modelos con los que cuenta YOLOv5 para crear el modelo. Una vez hecho todo lo anterior solo se ejecuta el código y comienza el entrenamiento de la red. Cuando el programa termina de ejecutar el entrenamiento devuelve un archivo PT el cual contiene el modelo de detección con pesos y sesgos para llevar a cabo la detección del objeto de interés. Este archivo se utiliza para realizar predicciones.

3.6.2.- Entrenamiento Haar Cascade

Para llevar a cabo el entrenamiento de un modelo para detección de objetos en Haar Cascade es necesario hacer uso de una herramienta de software conocida como Cascade-Trainer- GUI. El proceso de entrenamiento inicia importando la carpeta donde sen encuentran las imágenes positivas y negativas de interés para el modelo (ver figura 20). Posteriormente se requieren configurar dos parámetros: el porcentaje de imágenes positivas usadas y el número de imágenes negativas. El primer parámetro es el porcentaje de imágenes positivas usadas por época de entrenamiento, el segundo parámetro es el número total de imágenes negativas del conjunto de datos.

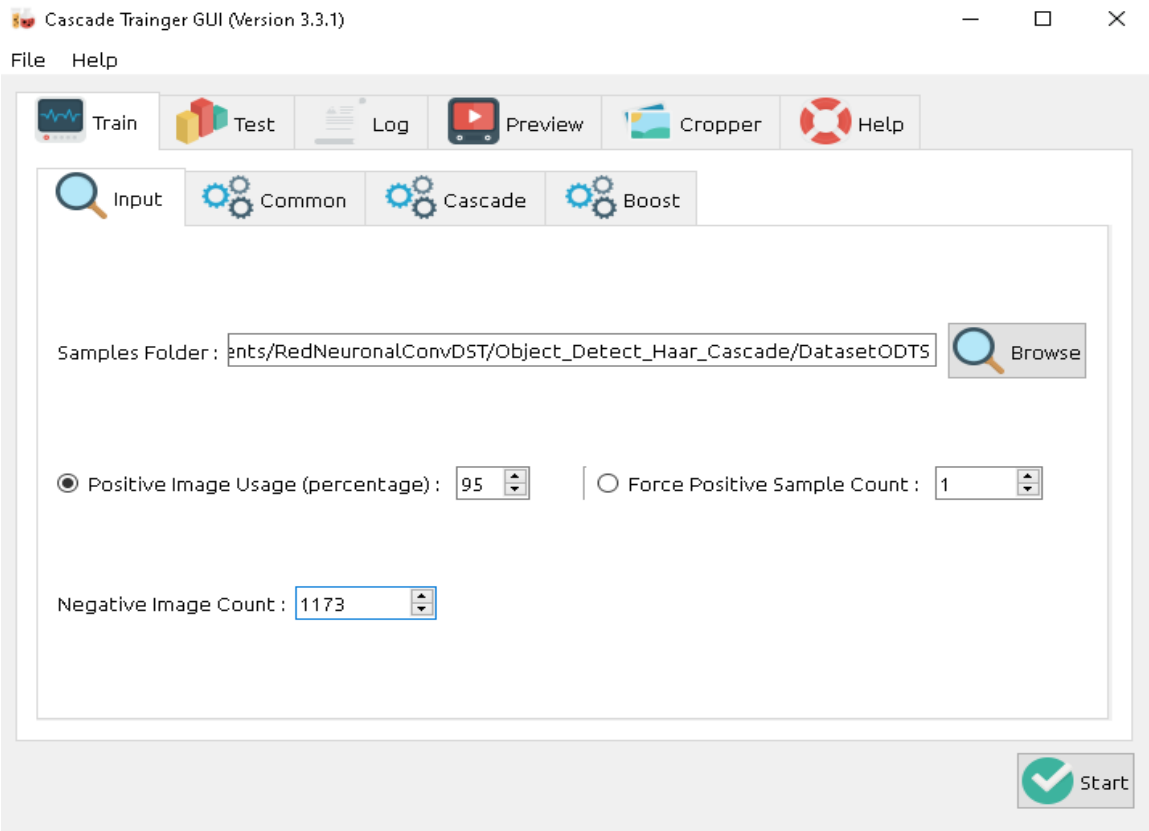

Figura 20.- Importar carpeta con el conjunto de datos en Cascade-Trainer GUI.

De acuerdo con la documentación oficial de Cascade-Trainer- GUI en la pestaña Common (ver figura 21) se requiere configurar el número de épocas o etapas de entrenamiento, así como el tamaño del Buffer de cálculo previo esto para ayudar con la velocidad de entrenamiento. Para dispositivos con 8 GB de RAM se recomienda trabajar de manera segura con un tamaño de Buffer de 2048 Mb. Los demás parámetros de esta pestaña se recomienda dejarlos con los valores predeterminados por software.

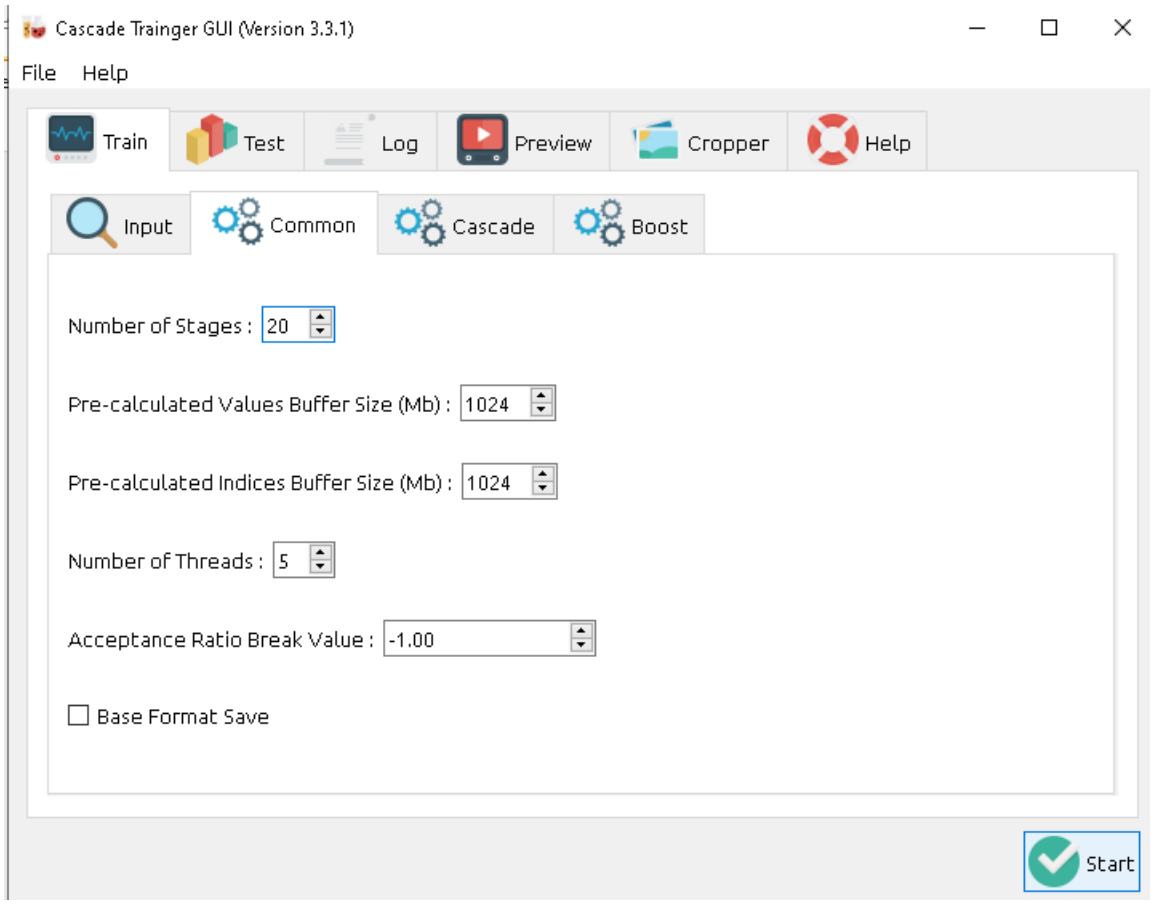

Figura 21.- Configuración de pestaña Common en Cascade-Trainer GUI.

En la pestaña Cascade (ver figura 22) se configuran los tamaños de las imágenes de muestra para el entrenamiento del modelo. La documentación sugiere no elegir tamaños muy grandes debido a que esto aumenta el tiempo de procesamiento en la detección de objetos. La configuración recomendada por el software para la altura y el ancho es de 24 lo que equivale a tener una imagen de 240x240 pixeles. Lo siguiente es definir la función Haar para el modelo, esta función requiere más tiempo de entrenamiento, pero tiene una tasa mayor de precisión.

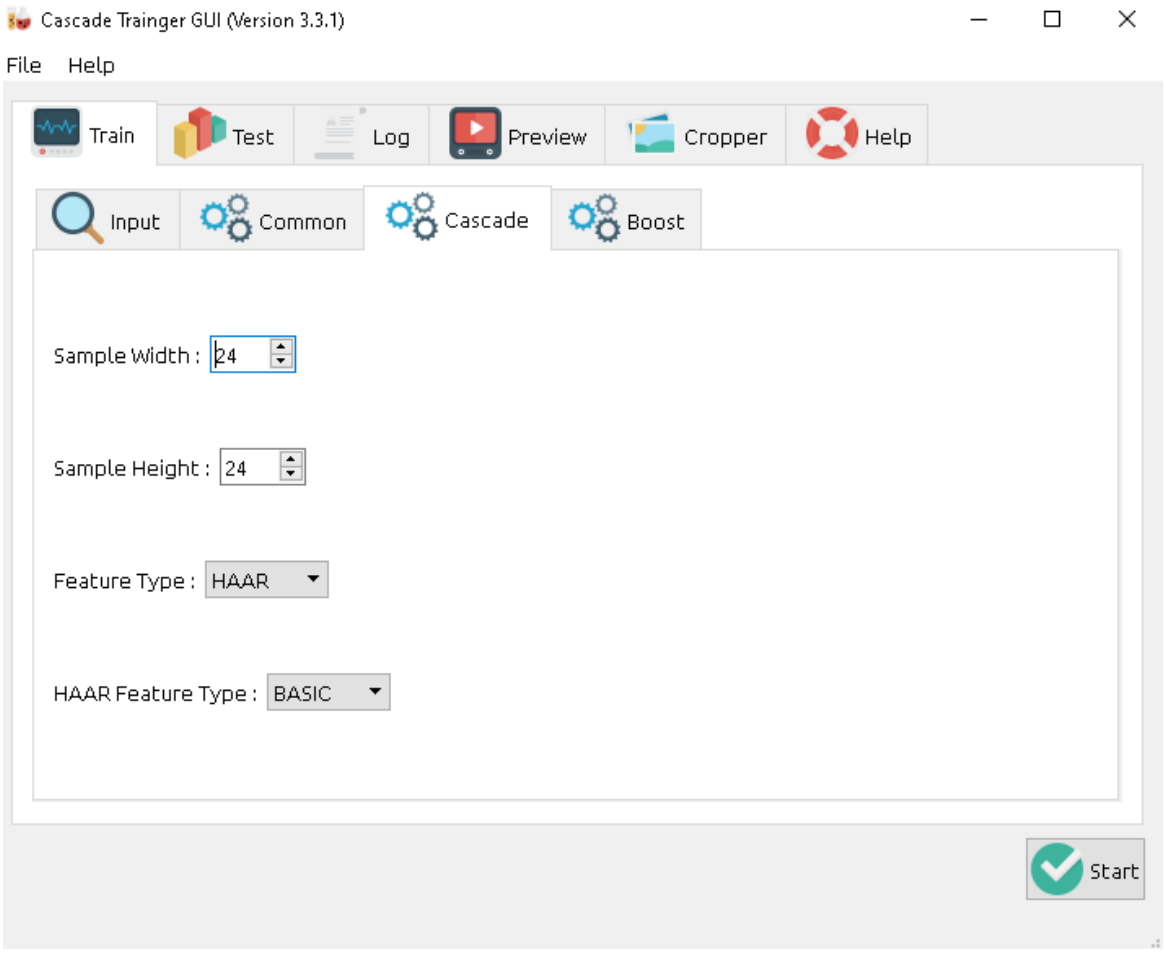

Figura 22.- Configuración de pestaña Cascade en Cascade-Trainer GUI.

Para la pestaña Boost (ver figura 23) del software la documentación recomienda dejar los parámetros con los valores predeterminados por el software. Una vez configurados todos los parámetros del modelo es siguiente paso es presionar el botón de inicio para que el software Cascade-Trainer GUI comience el entrenamiento del modelo de detección.

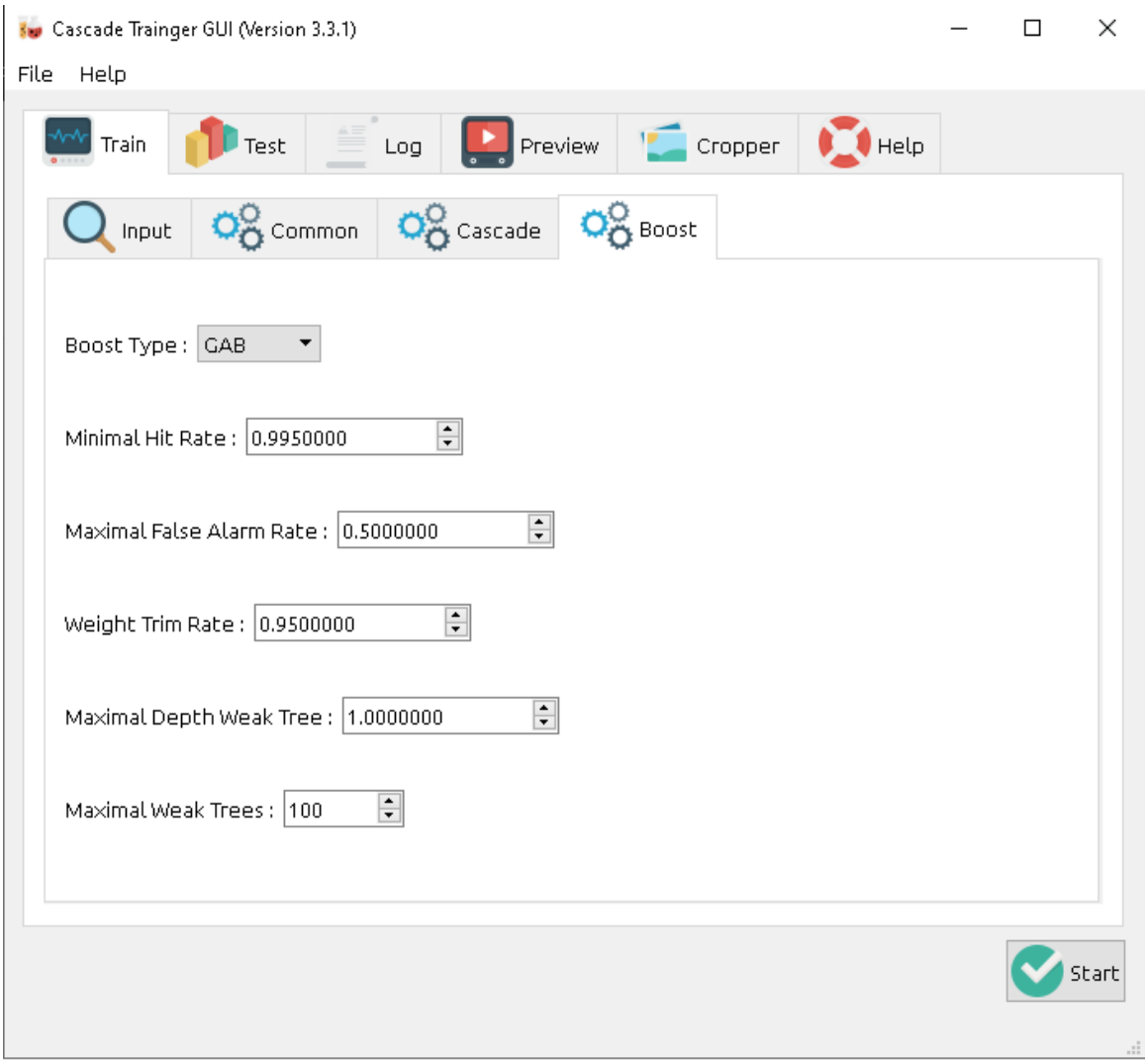

Figura 23.- Configuración de pestaña Boost en Cascade-Trainer GUI.

Al terminar el proceso de entrenamiento Cascade-Trainer GUI devuelve un archivo XML con el modelo de detección de objetos entrenado para detectar el objeto de interés. Este archivo será importado en un script en Python para realizar detecciones de señales mexicanas de tráfico.

3.7.- Detección de señales mexicanas de tránsito preventivas con modelos YOLOv5 y Haar Cascade

En esta sección se describe el proceso para llevar a cabo las predicciones en la detección de señalamientos de tránsito por los modelos YOLOv5 y Haar Cascade previamente entrenados. Ambos modelos (YOLOv5 y Haar Cascade) se importan a un script programado en Python.

3.7.1.- Predicciones con modelo YOLOv5

Para llevar a cabo predicciones en la detección de señalamientos mexicanos de tránsito con el modelo YOLOv5 previamente entrenado, se comienza con la creación de un script en Python. En este script se importa la librería cv2 (ver figura 24) de OpenCV, librería que se utiliza para el procesamiento de imágenes. También es necesario importar la librería torch con el fin de poder leer y cargar el modelo de detección YOLOv5. Al trabajar con imágenes se requiere el uso de la librería numpy ya que esta librería permite trabajar con operaciones de matrices de forma eficiente. Para el procesamiento de datos el programa se apoya de la librería pandas para su manejo y análisis. Otras librerías adicionales en este script se muestran en la figura 24.

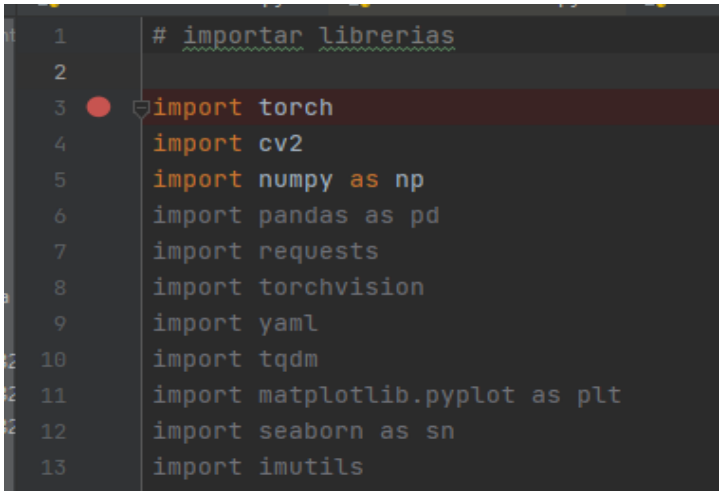

Figura 24.- Librerías utilizadas por el script de detección de señales de tráfico (modelo YOLOv5).

Una vez que el programa lee el modelo YOLOv5 lo siguiente es llevar a cabo la captura de video ya sea de una cámara webcam conectada al computador o bien de un video almacenado en el equipo de cómputo. La captura de video se realiza con un ciclo while el cual lee fotograma por fotograma. Después de la lectura del fotograma de la captura de video, el siguiente paso es implementar el modelo de detección sobre cada fotograma de la captura de video para que el modelo realice la detección de los objetos de interés en este caso particular de los señalamientos de tránsito. En este punto el modelo es capaz de detectar los objetos de interés en las imágenes que procesa. Lo que sigue es obtener la información del lugar en la imagen donde se encuentran los objetos detectados por el modelo, para ello se hace uso de una función de la librería pandas la cual obtiene las coordenadas en la imagen donde el objeto es detectado. Estas coordenadas forman un recuadro alrededor del objeto. Con esta información el recuadro que encierra el objeto detectado es recortado de la imagen y mostrado en una ventana.

3.7.2- Predicciones con modelo Haar Cascade

Con el modelo de detección Haar Cascade entrenado para la detección de señalamientos de tránsito el siguiente paso es realizar predicciones del modelo en una captura de video. Para llevarlo a cabo se crea un script en Python. En este caso en particular solo se requieren de dos librerías en el programa: cv2 para leer el modelo y numpy para las operaciones del procesamiento de imágenes de la captura de video (ver figura 25).

```
1  # Importar Librerias
2
3  import cv2
4  import numpy as np
```

Figura 25.- Librerías utilizadas por el script de detección de señales de tráfico (modelo Haar Cascade).

Al igual que el programa para predecir del modelo YOLOv5, el programa para realizar predicciones del modelo Haar Cascade también requiere realizar captura de video para realizar la detección de objetos, este proceso de captura de video también lo realiza con ciclo while leyendo cada uno de los fotogramas del video captura. Para leer el modelo Haar Cascade entrenado previamente se utiliza una función de la librería OpenCV para leer y cargar modelos de clasificadores de cascada. Dentro del ciclo while que realiza la captura de video se hace uso de una función de OpenCV para detectar objetos de multi escala, esta función combinada con el modelo Haar Cascade previamente entrenado detecta objetos de diferentes tamaños en una imagen de entrada. La función devuelve los objetos detectados como una lista de rectángulos. Estos rectángulos son las zonas de la imagen donde se encuentra el objeto detectado. Esta función proporciona las coordenadas del rectángulo de cada objeto detectado en la imagen. Con esta información se segmenta la porción de imagen del objeto detectado y se crea una nueva imagen con el objeto detectado. Esta nueva imagen se redimensiona y se muestra en una ventana nueva.

3.8.- Recolección de datos del modelo de clasificación de señales de tráfico

En esta etapa de desarrollo se lleva a cabo la recolección de datos (imágenes) para el entrenamiento de una red neuronal de convolución que clasifique cinco clases de señales diferentes. Las señales de interés para este proyecto son las mostradas en la figura 26. Se seleccionaron los cinco señalamientos de tránsito con mayor presencia en carreteras mexicanas.

Figura 26.- Señalamientos preventivos seleccionados para el estudio (SCT a, 2022).

El proceso para la adquisición de imágenes que contengan las señales de tránsito de interés se muestra en la figura 27. Se utilizan dos herramientas para obtener imágenes de señales de tráfico: Google Street View y el script de Python que realiza la detección de señales de tránsito con el modelo YOLOv5. Haciendo uso de Google Street View se obtienen imágenes de caminos y carreteras mexicanas que contengan las señales de tránsito de estudio. Cada una de estas imágenes es recortada (ver figura 27) con el fin de obtener una imagen donde solo se encuentre el objeto de interés. Este trabajo se hace de manera manual editando y recortando cada una de las imágenes extraídas de Google Street View. El siguiente paso es almacenar la imagen de la señal de tránsito en una carpeta que solo contenga imágenes con la misma señal de tráfico. Para este caso particular se cuenta con cinco carpetas para cada una de las clases de estudio. Cada carpeta lleva por nombre la etiqueta de las señales que contiene (ver figura 27). Las cinco carpetas se deben almacenar en una carpeta con el nombre de “SeTransito”, esta

carpeta es la que se utiliza como el conjunto de datos para el entrenamiento del modelo clasificador de señales de tránsito.

Figura 27.- Proceso de adquisición de imágenes para conjunto de datos de modelo clasificador de señales de tránsito.

Otra forma de obtener imágenes de señales de tráfico es utilizando el programa en Python desarrollado para realizar la detección y segmentación de señales de tránsito con el modelo YOLOv5 entrenado en la sección. A este programa solo es necesario importarle un video de carretera que contenga los señalamientos para el estudio. Debido a que este script ya realiza la detección y segmentación de señales de tránsito solo es necesario modificar el programa para que también almacene las imágenes de las señales detectadas en la carpeta del programa. Una vez que se realiza la modificación anterior al código el siguiente paso es eliminar las imágenes almacenadas que presenten distorsión en su imagen. El último paso es almacenar en la carpeta “SeTransito” cada una de las imágenes en la carpeta de la clase a la que pertenece la imagen.

3.9.- Diseño y programación del modelo de clasificación de señales de tráfico

En esta sección se describe el diseño de una red neuronal de convolución para la clasificación de imágenes en diferentes clases. La programación de la red se lleva a cabo en Python. Para llevar a cabo la programación de la red se utilizan dos librerías de código abierto como lo son: tensorflow y keras. Estas dos librerías trabajan en conjunto para crear y entrenar redes neuronales que detecten patrones en conjuntos de datos.

El primer paso es crear un script en Python para diseñar y compilar el modelo de la red neuronal convolucional clasificadora. Posteriormente se importan todas las librerías a utilizar en el programa (ver figura 28).

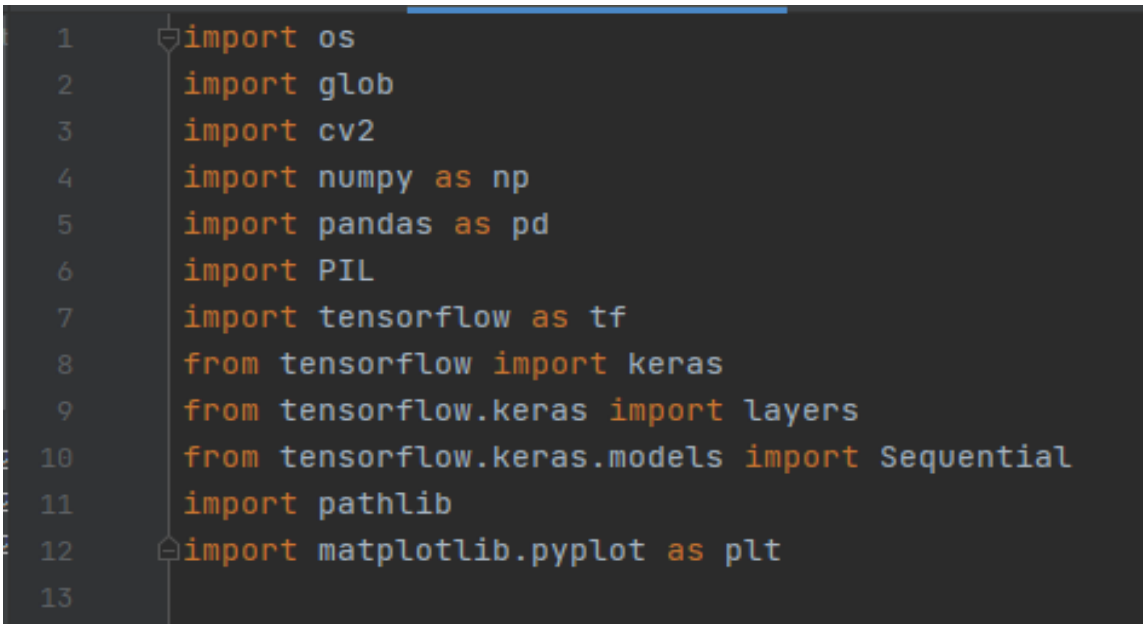

Figura 28.- Librerías de script del modelo de clasificación.

Para etiquetar el conjunto de datos de entrenamiento es necesario almacenar la carpeta “SeTransito” en la carpeta del script donde será programada la red neuronal y asignar la dirección de la carpeta en una variable (ver figura 29). Posteriormente se le aplica la función “pathlib.Path(data_set)” (línea 15 de la figura) a la variable donde se asigna la dirección del conjunto de datos de entrenamiento

esto con el fin de identificar los archivos que se encuentran en esta ruta que compartan una extensión o patrón determinado. La variable de la línea 17 sirve para asignar el número de ejemplos de mostrados a la red por iteración de aprendizaje. Las variables “img_height” y “img_width” determinan la altura y el ancho de las imágenes de entrenamiento y validación respectivamente.

```
13
14  data_set = "C:\\Users\\MAU\\Documents\\RedNeuronalConvDST\\ClasificadorMST\\SeTransito"
15  data_dir = pathlib.Path(data_set)
16
17  batch_size = 64
18  img_height = 180
19  img_width = 180
```

Figura 29.- Líneas de código para el etiquetado de los datos de entrenamiento.

Haciendo uso de una función de la librería keras (tf.keras.utils.image_dataset_from_directory()) se crean dos subconjuntos de datos: “training” (entrenamiento) y “validation” (validación), ver figura 30. El subconjunto “training” contiene los datos con los que el modelo será entrenado y el subconjunto “validation” sirve para validar el desempeño del modelo en cada iteración de entrenamiento. Para generar estos dos subconjuntos de datos se programan las líneas de la figura 30. Para este modelo de clasificación se utiliza el 70% del conjunto de datos para entrenamiento y 30% para validación. De esta forma se etiquetan todos los datos del conjunto de datos, la etiqueta que se le asigna a cada dato es el nombre de la carpeta donde se encuentra almacenado. El número de clases con las que trabaja el modelo es el número de carpetas guardadas en la carpeta del conjunto de datos en este caso particular se cuentan con cinco clases.

```
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.3,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)

val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.3,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)

class_names = train_ds.class_names
```

Figura 30.- Líneas de código para generar los subconjuntos de datos para el entrenamiento y validación del modelo.

El siguiente paso es crear el modelo de la red neuronal, para ello se programan las líneas de código de la figura 31, donde se definen el número de capas convolucionales de la red, para el modelo clasificador se utilizan 3 capas de convolución de 8,16 y 32 filtros por capa respectivamente, el tamaño del filtro o kernel para cada capa es convolucional es de 3. Las columnas y filas de alrededor de las matrices de convolución son rellenadas con ceros. La función de activación para cada una de las capas de convolución es la función “relu”. El modelo cuenta con 3 capas de reducción de dos dimensiones. Se utiliza la operación de Max Pooling para la reducción de datos de las matrices de las capas de convolución. En cada iteración de entrenamiento se desactivan el 20% del total de neuronas del modelo. Se agrega una capa de aplanamiento de capa. Dos capas densas una con que trabaja con la función “relu” y la última trabaja con la función de activación” softmax”. Esta última capa realiza una predicción para determinar el porcentaje que

tiene la entrada de pertenecer a cada una de las cinco clases con las cuales trabaja el modelo de clasificación.

```
46
47    # Crear Modelo
48    num_classes = len(class_names)
49
50    model = Sequential([
51      layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
52      layers.Conv2D(8, 3, padding='same', activation='relu'),
53      layers.MaxPooling2D(),
54      layers.Conv2D(16, 3, padding='same', activation='relu'),
55      layers.MaxPooling2D(),
56      layers.Conv2D(32, 3, padding='same', activation='relu'),
57      layers.MaxPooling2D(),
58      layers.Dropout(0.2),
59      layers.Flatten(),
60      layers.Dense(16, activation='relu'),
61      layers.Dense(num_classes, activation='softmax')
62    ])
63
64
```

Figura 31.- Líneas de código de la estructura de la red neuronal diseñada para la clasificación de imágenes.

Una vez que se termina de configurar el diseño del modelo lo que resta es compilar el modelo de la red neuronal (ver figura 32).

```
64
65    model.compile(optimizer='adam',
66                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
67                  metrics=['accuracy'])
68
```

Figura 32.- Líneas de código para compilar la red neuronal convolucional diseñada para la clasificación de imágenes.

3.10.- Entrenamiento de modelo de clasificación

El proceso de entrenamiento del modelo de clasificación es el que se muestra en la figura 33. A partir del conjunto de datos de entrenamiento recolectado se hace uso del modelo de red neuronal convolucional diseñado y programado en

Python para entrenar el modelo con el fin de obtener un archivo con extensión .h5 que contenga el modelo entrenado con los pesos y sesgos para realizar predicciones y clasificar las imágenes en una de las cinco clases seleccionadas para el caso de estudio.

Figura 33.- Proceso de entrenamiento de red neuronal convolucional para la clasificación de señales de tránsito preventivas mexicanas.

Para llevar a cabo el entrenamiento es necesario programar y compilar las líneas de código de la figura 34. En la línea 75 se define el número de épocas que el modelo será entrenado. Posteriormente se hace uso de la función “fit” para que el modelo de clasificación entrene durante 20 épocas. Con la función “history” se obtienen los valores del rendimiento de aprendizaje durante las distintas etapas de entrenamiento y con esto valores graficar las curvas de aprendizaje del modelo.

```
epochs=20
history = model.fit(train_ds,validation_data=val_ds,epochs=epochs)
# Visualizar los resultados del entrenamiento
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

# Guardar modelo
model.save("modelo_bs_64_vs_3_do_2_3CC81632_3CMP_1CD16_20epNS2.h5")
```

Figura 34.- Líneas de código para entrenar el modelo de clasificación.

Para poder graficar las curvas de aprendizaje del modelo una vez que termina el proceso de entrenamiento es necesario programar y compilar las líneas de código de la figura 35. Estas gráficas de desempeño sirven para evaluar y diagnosticar que tan bien aprende el modelo con el conjunto de datos de entrenamiento.

```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Exactitud de entrenamiento')
plt.plot(epochs_range, val_acc, label='Exactitud de validación')
plt.legend(loc='lower right')
plt.title('Exactitud de entrenamiento y vaidación')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Perdida de entrenamiento')
plt.plot(epochs_range, val_loss, label='Perdida de validación')
plt.legend(loc='upper right')
plt.title('Pérdida de entrenamiento y validación')
plt.show()
```

Figura 35.- Líneas de código para graficar las curvas aprendizaje del modelo de clasificación.

3.11.- Síntesis de modelos de detección (YOLOv5 y Haar Cascade) con modelo de clasificación

La última etapa de desarrollo del proyecto es enlazar los modelos de detección con el modelo de clasificación de tal modo que se obtenga un sistema capaz de detectar señalamientos de tránsito y a su vez determinar el tipo de señal detectada. El proceso de síntesis de los modelos de detección y el modelo de clasificación es el que se muestra en la figura 36. El modelo de clasificación trabaja para ambos modelos de detección.

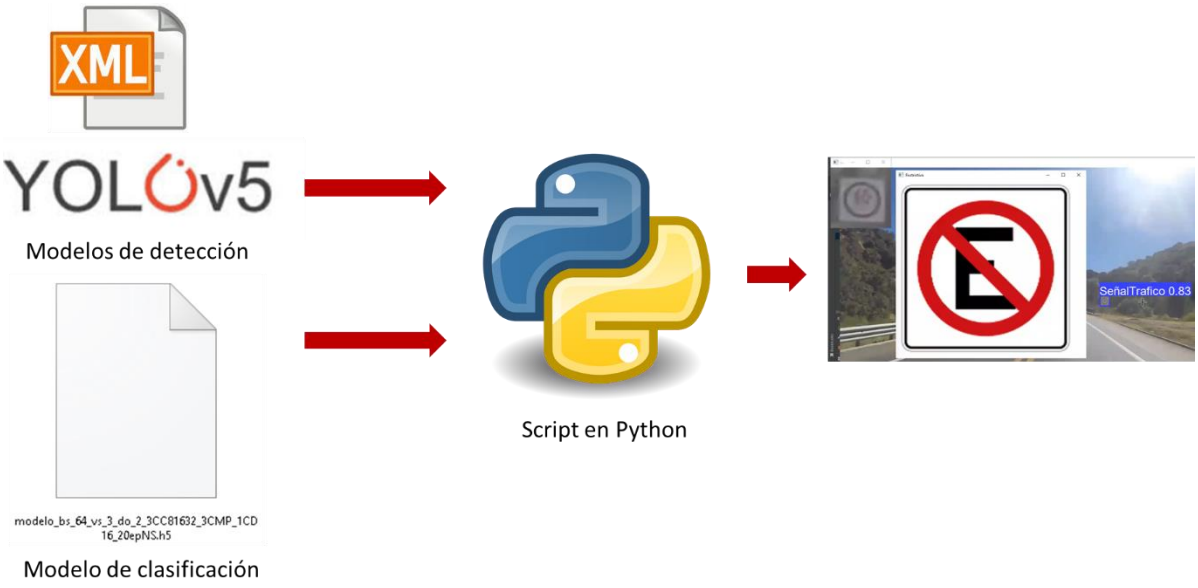

Figura 36.- Líneas de código para graficar las curvas aprendizaje del modelo de clasificación.

Para que los modelos de detección trabajen en conjunto con el modelo de reconocimiento es necesario enlazarlos a través de un script en Python. Para ahorrar tiempo de programación se toma como base el script en Python donde cada uno de los modelos de detección realiza predicciones debido a que estos scripts ya realizan la tarea de detectar señalamientos de tránsito y segmentar la imagen de la señal detectada en una nueva imagen. Lo que resta es leer el modelo de

clasificación en este script y realizar predicciones con las imágenes de señales de tránsito mexicanas preventivas que segmenta el modelo de detección. Cada una de estas imágenes es evaluada por el modelo de reconocimiento. El modelo de reconocimiento básicamente determina el porcentaje de similitud que tiene la imagen de entrada con cada una de las clases que clasifica el modelo. Si el porcentaje es superior al 95% en una de las clases el sistema muestra en una ventana el icono del señalamiento de tránsito que le pertenece a dicha clase (ver figura 36). Además de mostrar en una ventana el icono de la señal el sistema también ofrece un mensaje en consola con el nombre de la señal detectada.

3.12.- Consideraciones éticas del proyecto.

Todo el desarrollo de software estará sujeto a la familia de normas ISO/IEC 25000, en específico la ISO/IEC 25010 la cual describe el modelo de calidad para el producto software y para la calidad en uso. Además, el proceso para la toma de muestras (videos de carretera) para la matriz de pruebas se llevará a cabo con apego al reglamento de la ley tránsito para el estado de Querétaro. Todo el trabajo de experimentación se llevará de manera segura ya que las pruebas al sistema se llevarán a cabo con videos de carretera en un computador por lo que no existe riesgo alguno que afecte a terceros.

Capítulo 4
Experimentación y resultados

En este capítulo se presenta la metodología de experimentación, así como los resultados de la aplicación de la matriz de pruebas. Antes de dar paso a la experimentación primero se hace una revisión de las curvas de aprendizaje que se obtuvieron como resultado del entrenamiento de los modelos de detección y clasificación de señales mexicanas de tránsito preventivas. Posteriormente se aborda la puesta en experimento del proyecto para dar pie al análisis de los resultados de la matriz de pruebas. Por último, se presentan las conclusiones y prospectivas del proyecto.

4.1.- Curvas de aprendizaje de modelo de detección YOLOv5

Una vez que el modelo YOLOv5x termina el entrenamiento se obtienen los gráficos de desempeño a lo largo del entrenamiento. Estos gráficos son comúnmente conocidos como curvas de aprendizaje las cuales sirven para determinar qué tan bueno o que tan malo fue el aprendizaje del modelo durante el proceso de entrenamiento. La primera curva de aprendizaje es la de la figura 37 la cual indica las métricas de la precisión promedio media “mAP” (del inglés mean Average Precision).

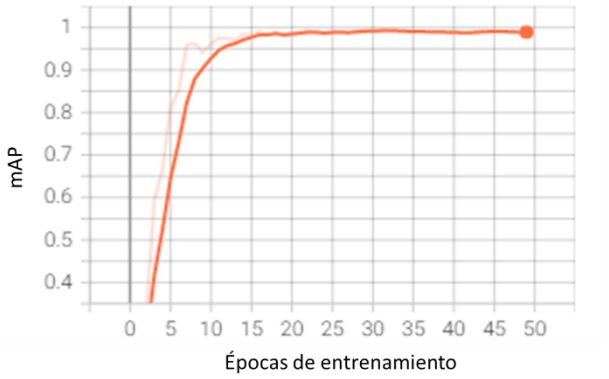

Figura 37.- Métricas de la precisión promedio media mAP.

Como se observa en la gráfica de la figura 37 el mAP con umbral de 0.5 alcanza un valor estable a partir de la época 15 de entrenamiento lo que indica que pasando esta época el modelo alcanza un porcentaje de más del 95% en su mAP

cuando las predicciones hechas por el modelo superan un umbral de 0.5. En la figura 38 se encuentra la gráfica de la curva de precisión media mAP 0.5:0.95 la cual evalúa el modelo de detección usando diferentes valores de umbral en un rango de 0.5 a 0.95.

Figura 38.- Gráfica de la curva de precisión media mAP 0.5:0.95 del modelo YOLOV5x.

La forma de la curva indica que la tasa de precisión del modelo es menor en comparación con la gráfica de la figura 37. Esta disminución en la tasa de precisión ocurre debido a que en la primera gráfica solo se trabaja con un umbral de 0.5 por lo que cualquier predicción que supere este valor de umbral se toma como un verdadero positivo debido a esto se alcanza una tasa de precisión alta. Caso contrario a la gráfica de la figura 38 ya que esta evalúa el modelo con distintos valores de umbral por lo que al ir aumentando este valor es menor la cantidad de predicciones que son verdaderos positivos por lo que la precisión promedio media (mAP) es menor al tener una cantidad mayor de falsos positivos. La gráfica también muestra que los valores de mAP no alcanza un valor estable en las 50 épocas de entrenamiento. Las métricas de precisión del modelo YOLOv5x se presentan en la gráfica de la figura 39. En esta gráfica la curva de precisión crece de forma importante en las primeras 15 épocas de entrenamiento posteriormente entran en una zona con ligeras variaciones, pero se mantiene en un rango entre el 95 y 100%.

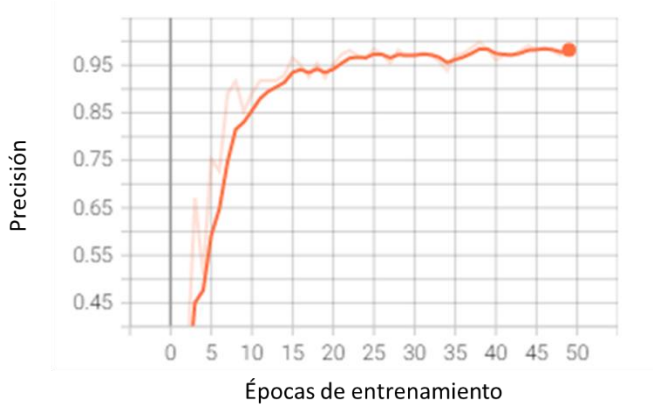

Figura 39.- Gráfica de la curva de precision del modelo YOLOv5x.

Otra métrica importante a analizar en el modelo YOLOv5x es la de recuperación la cual se muestra de forma gráfica en la figura 40. Esta devuelve el porcentaje de predicciones que el modelo identifica correctamente como pertenecientes a una clase de interés, para este caso particular de estudio el modelo YOLOv5 solo cuenta con una sola clase (señal de tránsito) por lo que la curva de la figura 40 muestra los porcentajes de probabilidad con los que el modelo detecta un objeto y lo clasifica como una señal de tráfico de manera correcta a lo largo de las 50 épocas de entrenamiento. Como se puede observar en la figura 39 el modelo supera el 90% de probabilidad de detectar y clasificar correctamente un objeto como señal de tráfico entre la novena y décima época de entrenamiento para después alcanzar un valor máximo del 97% entre las épocas 30 y 35 y finalmente mantenerse en un valor de alrededor del 95% en la última época de entrenamiento.

Figura 40.- Gráfica de la metrica de recuperación del modelo YOLOv5x.

La curva que se muestra en la gráfica de la figura 41 representa la perdida de caja en el entrenamiento del modelo YOLOv5x, esta métrica muestra los porcentajes de error del modelo al localizar el centro de un objeto y cubrirlo dentro de una caja delimitadora.

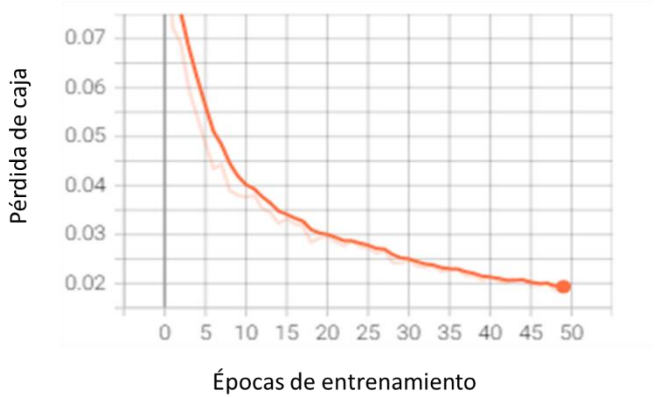

Figura 41.- Gráfica de la curva de perdida de caja de entrenamiento del modelo YOLOv5x.

En la figura 42 se presenta el gráfico de perdida de objetividad del modelo YOLOv5x a largo de las 50 épocas de entrenamiento, esta métrica es una medida de la tasa de probabilidad de que exista un objeto dentro de una zona de interés propuesta. Como se observa en el gráfico el porcentaje de error disminuye exponencialmente a largo del entrenamiento por lo que se puede decir que el

modelo detecta señales de tránsito en zonas de interés con porcentajes de error menores al 1% después de 8 épocas de entrenamiento.

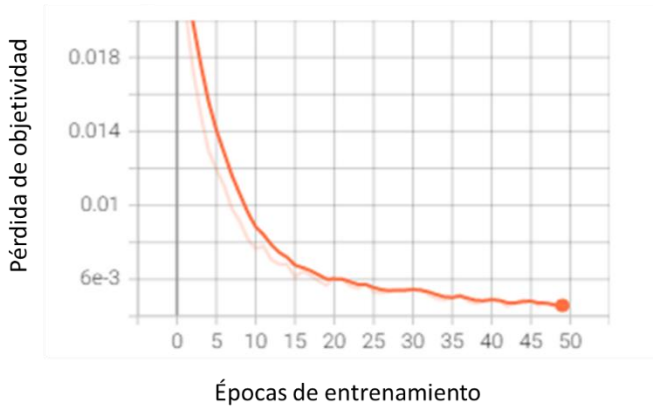

Figura 42.- Gráfica de la curva de perdida de objetividad de entrenamiento del modelo YOLOv5x.

La curva graficada en la figura 43 representa la curva de perdida de caja de validación del modelo YOLOv5x. Esta métrica al igual que la de la figura 40 muestra los porcentajes de error del modelo al localizar el centro de un objeto y cubrirlo en una dentro de una caja delimitadora. La diferencia radica en que esta gráfica de perdida de caja es del proceso de validación del modelo durante el entrenamiento de este. Como se observa en el gráfico el error en la detección del centro de un objeto de interés (señal de tráfico) en el modelo disminuye de forma exponencial hasta alcanzar un valor menor al 2% en la última época de entrenamiento.

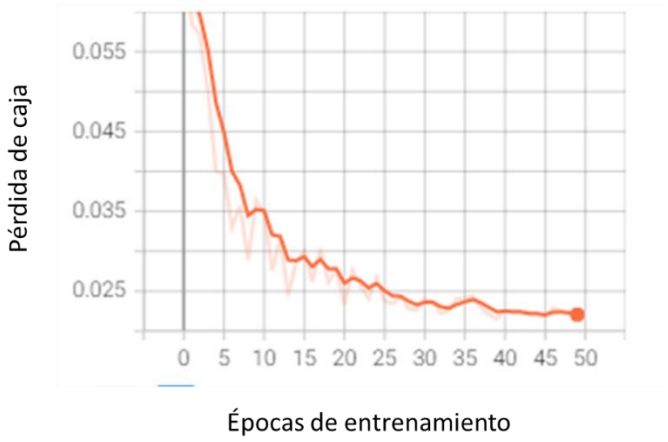

Figura 43.- Gráfica de la curva de perdida de caja de validación del modelo YOLOv5x.

Al igual que en la gráfica de la figura 41, la curva de la figura 43 representa la de perdida de objetividad del modelo YOLOv5x. Como se observa la curva disminuye de forma exponencial pero después de pasar la época 30 de entrenamiento se estabiliza en un valor por debajo del 1%.

Figura 44.- Gráfica de la curva de perdida de objetividad de validación del modelo YOLOv5x.

4.2.- Curvas de aprendizaje de modelo de clasificación

Una vez que el script donde se diseñó y programó la red neuronal de clasificación termina con el entrenamiento del modelo, el programa devuelve dos gráficas con las curvas de aprendizaje del modelo (ver figura 45 y 46). La gráfica de la figura 45 representa la tasa de exactitud de entrenamiento y validación del modelo en una escala de 0 a 1 en cada una de las 20 épocas de entrenamiento a las que fue sometido el modelo. La curva de color azul representa la exactitud del modelo en el entrenamiento (del inglés training accuracy) mientras que la de color naranja representa la exactitud del modelo en la etapa de validación (del inglés validation accuracy) de cada época de entrenamiento. Como se puede apreciar en la gráfica de la izquierda de la figura, el modelo de clasificación alcanza una estabilidad en su exactitud al pasar la décima época de entrenamiento mientras que la curva de validación alcanza un valor estable en su tasa de exactitud después de 15 épocas de entrenamiento. La curva de validación alcanza un valor menor de exactitud comparado con la curva de entrenamiento (ver figura 45).

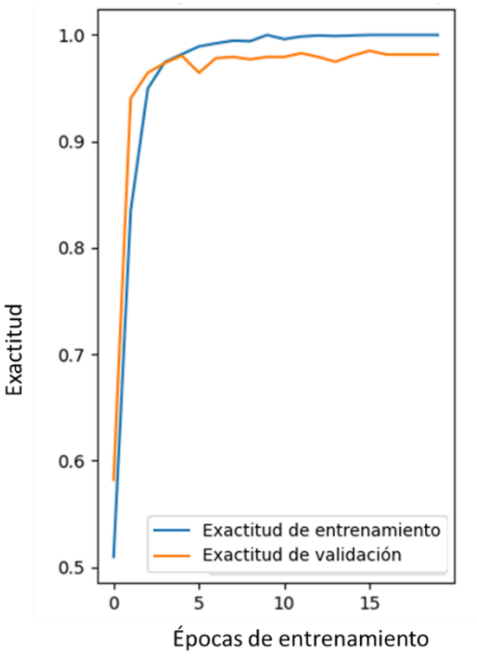

Figura 45.- Gráfica de las curvas exactitud del modelo de clasificación.

La gráfica de la figura 46 de la figura representa las curvas de pérdida o costo las cuales miden el error del modelo tanto en el entrenamiento como en la validación de este. Al igual que en la gráfica anterior, la curva azul representa la curva de entrenamiento y la naranja la de validación. De acuerdo con la gráfica de perdida la curva de entrenamiento continúa disminuyendo a medida que avanza la experiencia en el modelo. Además, la curva de validación disminuye hasta un punto entre las épocas 5 y 10 para posteriormente comenzar a subir levemente. Estas dos condiciones en las curvas de la gráfica de perdida son muestra de que existe un sobreajuste en el modelo.

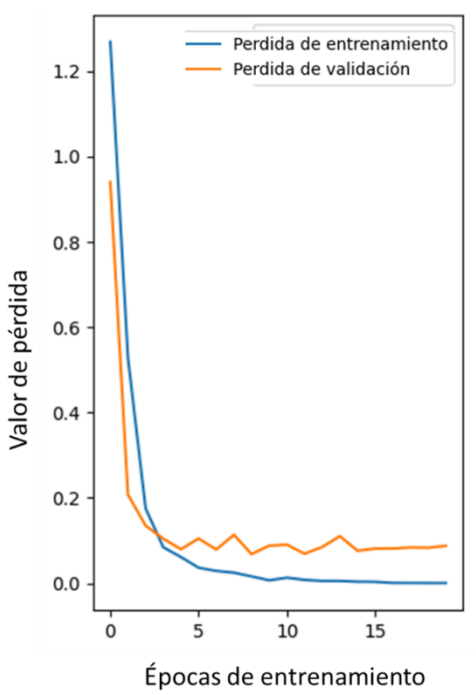

Figura 46.- Gráfica de las curvas de pérdida del modelo de clasificación.

Hay que aclarar que para el modelo de detección Haar Cascade no se obtuvieron métricas de aprendizaje debido a que el programa que se utilizó para su entrenamiento (Cascade-Trainer-GUI) no devuelve ninguna métrica de este tipo una vez que concluye el proceso de entrenamiento. Por lo que las únicas métricas para

este modelo serán las que se obtengan una vez que concluya la experimentación con la matriz de pruebas.

4.3.- Puesta en experimento

Para realizar las pruebas del sistema de reconocimiento de señales de tránsito se adquirieron un total de 200 videos de carretera. El proceso de adquisición de la toma de videos es el que se muestra en la figura 47. Para poder tomar las muestras se requiere de un vehículo en este caso en particular se hizo uso de un vehículo de la marca Volkswagen modelo Clásico del año 2012. A este vehículo se le colocó un base para celular en la zona del parabrisas (ver figura 47). Con un celular de la marca OPPO modelo A53 se tomaron las 200 muestras para el caso de estudio con una resolución de 1280x720. La suspensión del vehículo al momento de la toma de muestras se encontraba en condiciones nominales de operación por lo que no afecto a la toma de muestras para la experimentación.

Figura 47.- Proceso de adquisición de muestras para la matriz de pruebas.

Las 200 muestras (videos de carretera) se dividieron entre las 5 señales de tránsito de interés (curva derecha, curva izquierda, curva sinuosa, límite de altura y restrictiva) por lo que para cada señal se tomaron un total de 40 videos. Estos 40 videos por señal estaban divididos entre las 4 velocidades a evaluar en este proyecto (40, 60, 80 y 100 km/h) de tal manera que cada señal de tránsito a evaluar

en este caso de estudio contaba con 10 videos de prueba para cada una de las velocidades de la matriz de prueba.

4.4.- Matriz de pruebas

Para evaluar el sistema de reconocimiento de señales mexicanas de tránsito preventivas se hace uso de la matriz de pruebas de la tabla 1. Esta matriz de pruebas se aplica a cada una de las cinco señales de tránsito de estudio: curva a la derecha, curva a la izquierda, curva sinuosa, límite de altura y restrictiva. En total se realizan 200 pruebas a distintas velocidades de viaje para evaluar las tasas de detección y reconocimiento del sistema cuando se somete a cambios de velocidad de viaje. En el estudio se evalúan cuatro velocidades diferentes: 40, 60, 80 y 100 km/h. Cada señal de tránsito de interés se somete a 10 pruebas por velocidad. Las condiciones de iluminación, clima y nivel de desgaste en los señalamientos de tránsito que serán sujetos a prueba son los que se muestran en la tabla 1. Para todas las pruebas del estudio se considera que las señales de tránsito se encuentren completamente iluminadas. Todas las muestras recolectadas (videos de carretera) para llevar a cabo las pruebas serán tomadas en días soleados en un horario de entre las 12:00 y 14:00 horas del día. Solo señalamientos de tránsito que se encuentren en condiciones nominales de nivel de desgaste serán sometidos a prueba. Para la experimentación del sistema de reconocimiento se descartó el uso del modelo de detección YOLOv5x debido a que el tiempo de cómputo de dicho modelo es excesivamente elevado por lo que realizar pruebas es inviable para este modelo. Debido a lo anterior el sistema de reconocimiento de señales mexicanas de tránsito preventivas sometido a experimentación será el que utiliza el modelo Haar Cascade como modelo de detección de señales de tránsito.

Tabla 1.- Matriz de pruebas.

Detección de señales preventivas de tránsito					
Número de pruebas	Condiciones de iluminación	Condiciones climáticas	Nivel de desgaste en la señal de tránsito	Velocidad del vehículo	Horario de prueba
10	Completamente iluminado	Día soleado	Señal en condiciones nominales	40 km/h	12:00 a 14:00
10	Completamente iluminado	Día soleado	Señal en condiciones nominales	60 km/h	12:00 a 14:00
10	Completamente iluminado	Día soleado	Señal en condiciones nominales	80 km/h	12:00 a 14:00
10	Completamente iluminado	Día soleado	Señal en condiciones nominales	100 km/h	12:00 a 14:00

4.5.- Resultados

En esta sección se muestran los resultados de aplicar la matriz de pruebas a cada una de las señales de interés seleccionadas para este caso de estudio. Como ya se mencionó en la sección anterior, las pruebas al sistema de reconocimiento se aplicaron con los distintos videos capturados en la sección 4.3 los cuales se tomaron a cuatro velocidades diferentes: 40, 60, 80 y 100 km/h. La tabla 2 muestra los resultados de aplicar la matriz de pruebas a las cinco señales de tránsito de este proyecto. La cantidad de pruebas es de 10 pruebas por señalamiento. Para el algoritmo de detección se tienen dos columnas, una que contiene los verdaderos positivos es decir cuando el objeto detectado por el sistema es una señal de tráfico. La otra columna contiene los falsos positivos (cuando los objetos detectados por el sistema no son señales de tráfico). Por su parte el algoritmo de reconocimiento también cuenta con las columnas de verdaderos positivos y falsos positivos para los casos en donde el algoritmo clasifique de

manera correcta la señal detectada y cuando lo haga de manera incorrecta respectivamente.

Tabla 1.- Pruebas del sistema de reconocimiento de señales mexicanas de tránsito preventivas a 40 km/h.

Pruebas del sistema de reconocimiento de señales mexicanas de tránsito preventivas a 40 km/h					
Tipo de señalamiento	Cantidad de pruebas	Detección		Reconocimiento	
		Verdaderos positivos	Falsos positivos	Verdaderos positivos	Falsos positivos
Curva a la derecha	10	10	0	8	2
Curva a la izquierda	10	10	0	9	1
Curva sinuosa	10	10	0	0	10
Limite de altura	10	10	0	10	0
Restrictiva	10	9	1	9	0

Las pruebas del sistema de reconocimiento a una velocidad de desplazamiento de 40 km/h constantes muestran que la tasa de detección de señales de tránsito a esta velocidad de desplazamiento es del 98%. Sin embargo, lo que resalta de los resultados en la tabla 2 es que, de las 10 pruebas que se le aplicaron a la señal de curva sinuosa el algoritmo de reconocimiento no fue capaz de clasificar de manera correcta ninguna de las 10 pruebas debido a que la tasa de reconocimiento a 40 km/h es del 72%.

La figura 48 muestra al sistema de reconocimiento de señales mexicanas de tránsito preventivas detectar y reconocer una señal de tránsito preventiva de curva a la izquierda cuando el vehículo viajaba a una velocidad de 40 km/h. En la figura 48 se puede observar que una vez que el sistema de reconocimiento detecta y clasifica la señal de tránsito muestra un mensaje visual de asistencia a la conducción ya que en una ventana muestra el icono de la señal de tránsito detectada.

Figura 48.- Detección y reconocimiento de una señal de curva a la izquierda a una velocidad de desplazamiento de 40 km/h.

Las pruebas del sistema de reconocimiento a una velocidad de desplazamiento de 60 km/h constantes (ver tabla 3) muestran que la tasa de detección de señales de tránsito a esta velocidad de desplazamiento es del 96% lo cual indica una reducción del 2% comparado con la tasa de detección de las pruebas a 40 km/h constantes.

Tabla 2.- Pruebas del sistema de reconocimiento de señales mexicanas de tránsito preventivas a 60 km/h.

Pruebas del sistema de reconocimiento de señales mexicanas de tránsito preventivas a 60 km/h					
Tipo de señalamiento	Cantidad de pruebas	Detección		Reconocimiento	
		Verdaderos positivos	Falsos positivos	Verdaderos positivos	Falsos positivos
Curva a la derecha	10	10	0	6	4
Curva a la izquierda	10	10	0	8	2
Curva sinuosa	10	9	1	3	6
Limite de altura	10	10	0	8	2
Restrictiva	10	9	1	9	0

La tasa del algoritmo de reconocimiento también sufrió una reducción en comparación con las pruebas a 40 km/h. Para estas pruebas la tasa de

reconocimiento se redujo a 68%. En comparación con las pruebas anteriores en este caso el sistema de reconocimiento si pudo clasificar tres de las diez señales de prueba de la señal curva sinuosa. La figura 49 muestra la detección y reconocimiento de una señal de curva a la izquierda a una velocidad de desplazamiento del vehículo de 60 km/h.

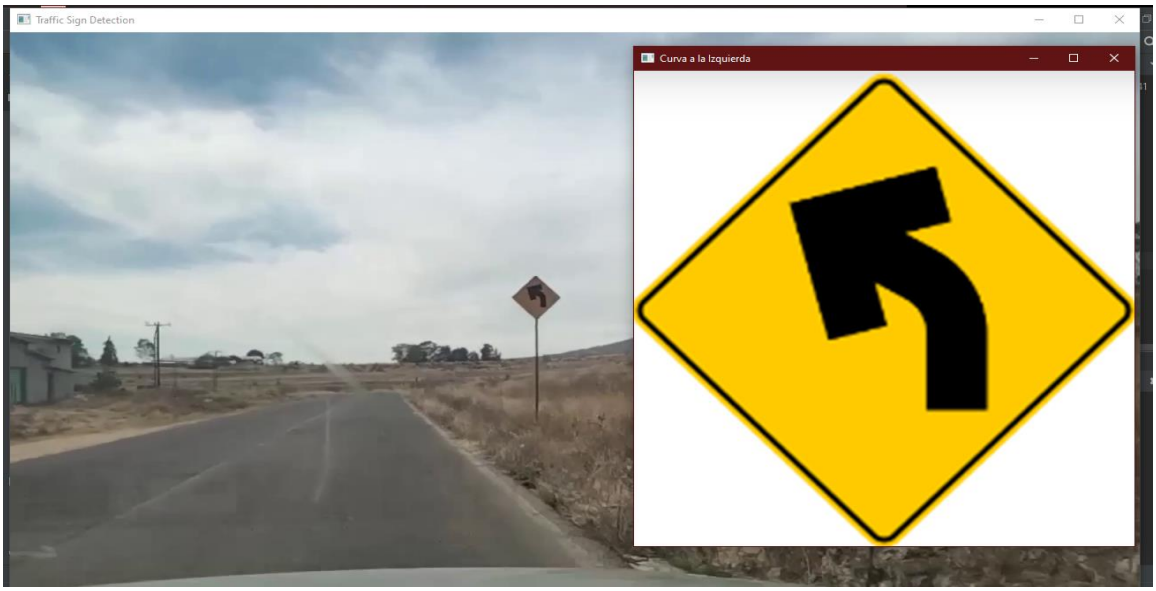

Figura 49.- Detección y reconocimiento de una señal de curva a la izquierda a una velocidad de desplazamiento de 60 km/h.

La tabla 4 contiene los resultados de las pruebas del sistema de reconocimiento de señales de tráfico a una velocidad constante de desplazamiento de 80 km/h. Siguiendo el patrón de las pruebas anteriores la tasa de exactitud de detección se redujo con el aumento de la velocidad, aunque en este caso la reducción fue más significativa ya que paso a tener un valor del 88% lo cual supone una reducción del 10% si lo comparamos con las pruebas a 40 km/h. El algoritmo de clasificación también siguió el patrón de reducción de su tasa de exactitud, aunque con una reducción más significativa comparada con el de detección, en este caso la tasa de exactitud en el reconocimiento de señales de tráfico fue del 56% lo cual es un 16% menos en comparación con las pruebas a 40 km/h.

Tabla 3.- Pruebas del sistema de reconocimiento de señales mexicanas de tránsito preventivas a 80 km/h.

Pruebas del sistema de reconocimiento de señales mexicanas de tránsito preventivas a 80 km/h					
Tipo de señalamiento	Cantidad de pruebas	Detección		Reconocimiento	
		Verdaderos positivos	Falsos positivos	Verdaderos positivos	Falsos positivos
Curva a la derecha	10	10	0	6	4
Curva a la izquierda	10	8	2	6	2
Curva sinuosa	10	8	2	1	7
Limite de altura	10	9	1	6	3
Restrictiva	10	9	1	9	0

La figura 50 muestra la detección y clasificación de una señal de tránsito preventiva de curva a la izquierda cuando el vehículo se desplaza a 80 km/h. Como se puede observar en la imagen de la figura 47 el sistema detecta la señal de tránsito preventiva y despliega el icono como un mensaje visual de asistencia a la conducción.

Figura 50.- Detección y reconocimiento de una señal de curva a la izquierda a una velocidad de desplazamiento de 80 km/h.

En la tabla 5 se enlistan los resultados de las pruebas hechas al sistema cuando el vehículo se desplazaba a 100 km/h. En este caso la tasa de exactitud se mantuvo igual que en las pruebas hechas a 80 km/h. Para el algoritmo de reconocimiento este incremento su porcentaje de exactitud de clasificación de señales de tráfico paso de 56% en las pruebas a 80 km/h a 58%.

Tabla 4.- Pruebas del sistema de reconocimiento de señales mexicanas de tránsito preventivas a 100 km/h.

Pruebas del sistema de reconocimiento de señales mexicanas de tránsito preventivas a 100 km/h					
Tipo de señalamiento	Cantidad de pruebas	Detección		Reconocimiento	
		Verdaderos positivos	Falsos positivos	Verdaderos positivos	Falsos positivos
Curva a la derecha	10	10	0	6	4
Curva a la izquierda	10	10	0	9	1
Curva sinuosa	10	6	4	1	5
Limite de altura	10	10	0	5	5
Restrictiva	10	8	2	8	0

La figura 51 muestra la detección y clasificación de una señal de tránsito preventiva de curva a la izquierda cuando el vehículo se desplaza a 100 km/h. Al igual que en las pruebas anteriores el sistema fue capaz de detectar la señal, clasificarla y mostrar un mensaje visual de asistencia a la conducción desplegando en una ventana el icono de la señal detectada.

Figura 51.- Detección y reconocimiento de una señal de curva a la izquierda a una velocidad de desplazamiento de 100 km/h.

4.6.- Características finales del sistema de reconocimiento de señales mexicanas de tránsito preventivas

En la tabla 6 se muestran las características finales del sistema de reconocimiento. De acuerdo con los resultados de la aplicación de la matriz de pruebas el modelo de detección de señales de tránsito disminuye su tasa de detección de señales a medida que la velocidad en el vehículo incrementa como se observa en la tabla 6. De contar con una tasa del 98% en la detección de señales a una velocidad de 40 km/h pasa a un 88% cuando la velocidad supera los 80 km/h lo cual representa una pérdida del 10% en su tasa de exactitud. La menor reducción se presenta cuando el vehículo pasa de 40 km/h a 60 km/h ya que está perdida es de solo el 2%.

Tabla 5.- Características finales del sistema de reconocimiento de señales mexicanas de tránsito preventivas.

Características finales del sistema		
Velocidad del vehiculo	**Tasa de detección**	**Tasa de reconocimiento**
40 km/h	**98%**	**72%**
60 km/h	**96%**	**68%**
80 km/h	**88%**	**56%**
100 km/h	**88%**	**58%**

En la parte de reconocimiento su tasa de exactitud es menor en comparación con la detección de señales ya que el valor máximo alcanzado fue de 72% cuando el vehículo viaja a 40 km/h. También se observa un comportamiento similar al algoritmo de detección ya que a medida que la velocidad del vehículo aumenta la tasa de exactitud del modelo de reconocimiento decrece. La menor tasa de reconocimiento se presentó cuando el vehículo viaja a 80 km/h con un valor del 56% (ver tabla 6).

4.7.- Conclusiones y prospectivas

En este proyecto de tesis se desarrolló un sistema de reconocimiento de señales mexicanas de tránsito preventivas el cual es capaz de detectar y reconocer cinco clases de señales de tránsito. Se desarrolló una base de datos con 8865 imágenes de señales de tránsito mexicanas, de las cuales 5953 imágenes se utilizaron para el entrenamiento de los modelos de detección y las 2912 imágenes restantes para el entrenamiento del modelo de clasificación. Para la parte de detección de señalamientos de tránsito se hizo uso de dos técnicas de aprendizaje de máquinas para la detección de objetos: redes neuronales convolucionales y clasificadores en cascada. Se entrenó un modelo a partir de una red neuronal YOLOv5x capaz de detectar señalamientos de tránsito mexicanos con una tasa de exactitud en la similitud de los objetos detectados con señales de tránsito superior al 75% en videos de carretera. Adicionalmente se entrenó un modelo de detección

de señales de tráfico utilizando un clasificador en cascada Haar Cascade. Por su parte se desarrolló y entrenó un modelo clasificador basado en una red neuronal de convolución capaz de clasificar 5 clases de señales de tránsito preventivas mexicanas.

El software desarrollado para el sistema cumple con 4 de las ocho características de la norma ISO/IEC 25010 las cuales son: adecuación funcional, compatibilidad, portabilidad y usabilidad. El software cumple con las funciones requeridas por el usuario. Hasta el momento realiza detecciones en señales de tránsito con una tasa de exactitud superior al 75% en videos de carretera. La programación es en lenguaje Python por lo que es portable para cualquier equipo que tenga instalado Python. Debido a que se hizo uso de herramientas de código abierto no existen restricciones de licencias para su uso. El software es compatible con otros equipos de hardware, en este caso específico es capaz de recibir datos de un equipo externo como lo es una webcam. El uso del software es bastante intuitivo por lo que cumple con la característica de usabilidad de la norma ISO/IEC 25010.

El modelo de detección de la red YOLOv5x mostró una mayor tasa de detección ya que no presenta falsos positivos en la detección de señales de tránsito a comparación del modelo Haar Cascade que si llega a presentar falsos positivos. En tiempo de procesamiento el modelo Haar Cascade fue superior al modelo YOLOv5x debido a que el ultimo tarda 11 minutos en procesar un video con 5 segundos de duración. Mientras que el modelo Haar Cascade tarda 18 segundos en procesar el mismo video. El excesivo tiempo de procesamiento del modelo YOLOv5x fue motivo suficiente para descartarlo de las pruebas realizadas al sistema de reconocimiento de señales de tráfico.

Tanto la tasa de detección como la de reconocimiento disminuyen a medida que la velocidad del vehículo incrementa. El modelo de detección presentó mayor porcentaje en su tasa de exactitud comparada con el modelo de reconocimiento. El modelo de detección de señales de tránsito alcanzo un valor máximo de 98% cuando el vehículo viajaba a 40 km/h y una mínima de 88% cuando el automóvil se

desplazaba a 100 km/h. En el caso del modelo de reconocimiento el valor máximo alcanzado en su tasa de exactitud al momento de determinar a qué clase pertenecían las señales de prueba fue del 72% cuando el automóvil se desplazaba a una velocidad constante de 40 km/h y una mínima del 56% cuando el vehículo viaja a 80 km/h constantes.

Aunque el sistema de reconocimiento arroja tasas de detección y reconocimiento de señales mexicanas de tránsito preventivas por encima del 70% a bajas velocidades (superior al 90% para el modelo de detección) se requiere el realizar un mayor de número de pruebas del sistema y con una mayor variedad de velocidades de desplazamiento del vehículo para validar que el sistema es capaz de detectar y reconocer señalamientos de tránsito en un rango más amplio de velocidades de desplazamiento y no solo en las planteadas en la matriz de pruebas de este proyecto.

Como prospectivas para el sistema de reconocimiento de señales mexicanas de tránsito preventivas desarrollado en este proyecto de tesis se propone el desarrollar un base de datos que abarque toda la colección de señales de tránsito mexicanas de la Secretaría de Comunicaciones y Transportes. Someter el sistema de reconocimiento de señales de tránsito a pruebas bajo diferentes condiciones de iluminación. Implementar el sistema de reconocimiento de señalamientos de tránsito en un sistema embebido. Entrenar un modelo para la clasificación de señales de tránsito que abarque todas las señales de tránsito mexicanas que contiene el banco digital de señalización vial de la Secretaría de Comunicaciones y Transportes. Evaluar la resolución del modelo cuando la calidad de los señalamientos se degrada. Implementar el sistema de reconocimiento de señales mexicanas de tránsito en un sistema embebido el cual pueda ser instalado en cualquier vehículo del parque vehicular mexicano. Hacer uso de técnicas de optimización para mejorar el tiempo de cómputo y los porcentajes de exactitud del sistema de reconocimiento.

Bibliografía

A. Moreno, E. Armengol, J. Béjar, L. Belanche, U. Cortes, R. Gavaldá, J.M. Gimeno, B. López, M. Martin, M. Sánchez. (1994). Aprendizaje automático. Edicions UPC.

Arévalo, Vicente & González-Jiménez, Javier & Ambrosio, Gregorio. (2004). La Librerí-a de Visión Artificial OpenCV. Aplicación a la Docencia e Investigación (in spanish). Base Informática. 40. 61-66.

Ayachi, R., Afif, M., Said, Y. et al. (2020). Traffic Signs Detection for Real-World Application of an Advanced Driving Assisting System Using Deep Learning. Neural Process Lett 51, 837–851.

B. Novak, V. Ilić and B. Pavković. (2020). YOLOv3 Algorithm with additional convolutional neural network trained for traffic sign recognition. Zooming Innovation in Consumer Technologies Conference (ZINC), pp. 165-168.

Benítez R, Escudero G, Kanaan S, Masip D. (2013) Inteligencia Artificial Avanzada. Ed UOC. ISBN: 978-84-9029-887-9

Berzal, F. (2018). Redes Neuronales & Deep Learning. Edición independiente.

de Jesús Ávila, I. (2020). Red neuronal convolucional para detección y clasificación de señales de tránsito. Mérida, Yucatán. Tesis de licenciatura. Universidad Autónoma de Yucatán.

Calvo, D. (2017). Red Neuronal Convolucional CNN. (https://www.diegocalvo.es/red-neuronal-convolucional/).

Calvo, D. (2018). Función de activación – Redes neuronales. (https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/).

Domínguez, T. (2021). Visión artificial: Aplicaciones con OpenCV-Python.

Esqueda, J.J. (2002). Fundamentos de Procesamiento de Imágenes. Evento: CONATEC 2002.

Farag, Wael. (2018). Recognition of Traffic Signs by Convolutional Neural Nets for Self-driving Vehicles. 205 – 214.

Ford. (2022). Sistema de reconocimiento de señales de tráfico (https://www.ford.es/compra/explora/tecnologia/experiencia-de-conduccion/sistema-de-reconocimiento-de-senales-de-trafico#:~:text=El%20sistema%20de%20reconocimiento%20de,zona%20con%20un%20l%C3%ADmite%20distinto.).

Gracia, A. (2012). Inteligencia Artificial: fundamentos, practica y aplicaciones. Alfaomega.

García, E. (2019). Introducción a las redes neuronales convolucionales. Aplicación a la visión por ordenador. Trabajo de tesis. Universidad de Zaragoza.

Giménez, J., Monsoriu, J.A., Alemany, E. (2016). Aplicación de convolución de matrices al filtrado de imágenes. Modelling in Science Education and LearningVolume 9(2).

Gutta, S. (2021). Object Detection Algorithm-YOLOv5 Architecture. (https://medium.com/analytics-vidhya/object-detection-algorithm-yolo-v5-architecture-89e0a35472ef).

INEGI. (2022). Accidentes de tránsito (https://www.inegi.org.mx/temas/accidentes/).

INSP. (2020). Exceso de velocidad en México (https://www.insp.mx/avisos/exceso-de-velocidad-en-mexico).

Jeremías, E. (2020). Reconocimiento de objetos a través de la metodología Haar Cascades.

Jet Brains. (2022). Vista General de PyCharm. (https://www.jetbrains.com/help/pycharm/quick-start-guide.html).

Jocher, G. (2020) Documentación de red YOLOv5 (https://docs.ultralytics.com/).

Jung, H. K., & Choi, G. S. (2022). Improved YOLOv5: Efficient Object Detection Using Drone Images under Various Conditions. Applied Sciences, 12(14), 7255.

Mercedes-Benz. (2022). Asistente para señales de tráfico (https://www.mercedes-benz.es/passengercars/mercedes-benz-cars/models/eqc/safety.pi.html/mercedes-benz-cars/models/eqc/safety/driving-assistance-gallery/traffic-sign).

M. M. William et al. (2019) Traffic Signs Detection and Recognition System using Deep Learning. Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), pp. 160-166.

OMS. (2022). Traumatismos causados por el tránsito (https://www.who.int/es/news-room/fact-sheets/detail/road-traffic-injuries).

Palma, J.T., Marín, R. (2008). Inteligencia Artificial: métodos, técnicas y aplicaciones. Mc Graw Hill.

Pineda, C.M. (2021). Aprendizaje automático y profundo en Python: Una mirada hacia la inteligencia artificial.

Ponce, P. (2010). Inteligencia artificial con aplicaciones a la ingeniería. Alfaomega.

Python. (2022). Página de inicio de Python. (https://www.python.org/).

RAE. (2022). Definición de Inteligencia (https://dle.rae.es/inteligencia).

Renault. (2022). Detección de paneles de señalización (https://es.e-guide.renault.com/esp/Captur-2/DETECCION-DE-PANELES-DE-SENALIZACION).

Rozada, S. (2021). Estudio de la arquitectura YOLO para la detección de objetos mediante Deep Learning. Valladolid. Tesis de Maestría. Universidad de Valladolid.

Rodríguez, R. C., Carlos, C. M., Vergara-Villegas, O. O., & Sánchez, V. G. C. (2020). Detección y clasificación de señales de tráfico mexicanas mediante aprendizaje profundo. Res. Comput. Sci., 149(8), 435-449.

SCT a. (2022). Banco Digital de Señalización Vial. (http://www.sct.gob.mx/bancodigital/)

SCT b. (2022). Dirección General de Conservación de Carreteras, Sección Señalamiento. (https://www.sct.gob.mx/carreteras/direccion-general-de-conservacion-de-carreteras/publicaciones/senalamiento/).

SCT c. (2022). Norma Oficial Mexicana NOM-034-SCT2-2011 “SEÑALAMIENTO HORIZONTAL Y VERTICAL DE CARRETERAS Y VIALIDADES URBANAS”. (https://www.sct.gob.mx/fileadmin/DireccionesGrales/DGST/Normas_Oficiales_Mexicanas/NOM-034-SCT2-2011.pdf)

Sánchez, A.J., Ricolfe, C. (2016). Sistema de Captura de Imágenes. Universidad Politécnica de Valencia.

Tesla. (2022). Control de semáforos y señales de stop (https://www.tesla.com/ownersmanual/modely/es_es/GUID-A701F7DC-875C-4491-BC84-605A77EA152C.html).

Treviño, G. (2020). Algoritmo para la detección de vehículos y peatones combinando CNN´s y técnicas de búsqueda. Querétaro, Querétaro. Tesis de Maestría. Universidad Autónoma de Querétaro.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* (Vol. 1, pp. I-I). Ieee.

Y. Sun, P. Ge and D. Liu. (2019) Traffic Sign Detection and Recognition Based on Convolutional Neural Network. Chinese Automation Congress (CAC), 2019, pp. 2851-2854.