EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CULIACÁN

SISTEMA GENERADOR DE INFORMACIÓN VIAL A PARTIR DE SECUENCIA DE IMÁGENES

TESIS

PRESENTADA ANTE EL DEPARTAMENTO ACADÉMICO DE ESTUDIOS DE POSGRADO DEL INSTITUTO TECNOLÓGICO DE CULIACÁN EN CUMPLIMIENTO PARCIAL DE LOS REQUISITOS PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

POR:

C. MARIO ALBERTO ROMÁN GARAY

ASESOR:

DR. HÉCTOR RODRÍGUEZ RANGEL

CULIACÁN, SINALOA Agosto del 2022

Declaración de autenticidad

Por la presente declaro que, salvo cuando se haga referencia específica al trabajo de otras personas, el contenido de esta tesis es original y no se ha presentado total o parcialmente para su consideración para cualquier otro título o grado en esta o cualquier otra Universidad. Esta tesis es resultado de mi propio trabajo y no incluye nada que sea resultado de algún trabajo realizado en colaboración, salvo que se indique específicamente en el texto.

Mario Alberto Roman Garay. Culiacán, Sinaloa, México, 2022.

Agradecimientos

Agradezco principalmente a **mi madre** que a lo largo de los años siempre ha sido mi principal motivación en mi vida personal y profesional, sin su amor, apoyo y consejos esto no habría sido posible. Ante cualquier adversidad, siempre ha sabido como salir adelante y brindarme los medios para mi desarrollo como persona.

A **mi familia** por el amor y apoyo que me han brindado desde que era pequeño.

Al **departamento de Maestría en Ciencias de la Computación** por brindarme todo lo necesario para llevar a cabo mi desarrollo profesional dentro de la institución.

Al **Dr. Héctor Rodríguez Rangel** por ser un excelente guía durante todo 1 año de trabajo de tesis, siempre aconsejando y apoyando en cualquier situación.

A **los profesores** de la Maestría por ser unos excelentes mentores dentro y fuera de clases.

Al **Instituto Tecnológico de Culiacán** por ser una institución a la vanguardia de la educación y apoyarme para salir adelante durante mi carrera universitaria y ahora en mi camino de postgrado.

A **Consejo Nacional de Ciencia y Tecnología** por el financiamiento y apoyo durante los 2 años de maestría.

A **mis compañeros** por estar siempre apoyando a pesar de las situaciones adversas de pandemia en las que estuvimos envueltos.

Dedicatoria

Dedico este trabajo a mi madre, quien durante toda mi vida ha luchado por darme todo lo necesario para mi desarrollo como persona y profesional. Para ti MAMÁ.

Resumen

El aumento de la población en las ciudades ha provocado un incremento en distintas problemáticas para sus habitantes. Algunas de los problemas son los accidentes vehiculares, mayor tiempo en el tráfico y embotellamientos. Ante estas problemáticas, los departamentos de tránsito necesitan recolectar información de los vehículos para realizar estudios de tráfico. Lo anterior para tomar decisiones con el objetivo de buscar rutas alternas, mejorar la infraestructura de carreteras, el uso de semáforos o construcción de nuevas vías de movilidad. Para ello utilizan distintas tecnologías, en algunos casos con costos elevados, además de recolectar características limitadas.

Este trabajo muestra el desarrollo de un sistema de extracción de información de tráfico vial, a partir de un archivo de video. Este es capaz de utilizar las cámaras de videovigilancia que se encuentran instaladas en distintos puntos de las ciudades, ahorrando así un gasto en la inversión de la puesta en marcha del sistema.

El sistema realiza detecciones, seguimiento, clasificación y conteo de vehículos. Para la tarea de detección y clasificación se utilizan las redes convolucionales de YOLOv3 junto a un conjunto de datos personalizado de 21,000 imágenes repartidas en las siguientes clases: auto, autobús, camioneta, van, motocicleta, bicicleta y persona. Para la tarea de seguimiento y conteo se utiliza la herramienta *AutoTrack* que implementa el filtro de Kalman y un algoritmo húngaro de seguimiento. Esta herramienta cuenta con una función de manejo de obstrucciones y reaparición de vehículos. A cada vehículo se le asigna un identificador, en caso de ocultarse y volver a aparecer se identifica como un objeto anteriormente detectado y no se suma al contador. En la implementación del sistema, el tiempo de procesamiento y extracción de características es de 1.5 segundos por cada segundo del archivo de video. El modelo que se entrenó a partir de un conjunto de datos personalizados logró alcanzar hasta un 95.43 % de precisión para clasificar. Los resultados obtenidos son equiparables a los existentes actualmente en las tareas de detección y clasificación; sin embargo, estos no cuentan con el manejo de excepciones, siendo esta diferencia la principal con ellos.

Palabras Clave

- Aprendizaje máquina
- Aprendizaje profundo
- Entrenamiento de modelos de clasificación
- Inteligencia artificial
- Redes neuronales convoluciones
- Sistemas Inteligentes de Transporte (ITS)
- Vehículo
- Visión artificial

Índice general

Índice de figuras VIII

Índice de tablas X

1. Introducción 1
- 1.1. Definición del problema . . . 4
- 1.2. Hipótesis . . . 6
- 1.3. Objetivo . . . 6
- 1.4. Justificación . . . 7
- 1.5. Estructura de la tesis . . . 8

2. Marco Teórico 10
- 2.1. Inteligencia Artificial . . . 10
- 2.2. Aprendizaje máquina . . . 11
- 2.3. Redes Neuronales . . . 12
  - 2.3.1. Redes multicapa . . . 13
  - 2.3.2. Propagación hacia atrás . . . 13
- 2.4. Visión artificial . . . 14
- 2.5. Aprendizaje por transferencia (*Transfer Learning*) . . . 15
- 2.6. Aprendizaje profundo . . . 15
  - 2.6.1. Arquitectura de redes profundas . . . 18
  - 2.6.2. Hiperparámetros . . . 18
    - 2.6.2.1. Tasa de aprendizaje . . . 18
    - 2.6.2.2. Funciones de activación . . . 19
    - 2.6.2.3. Funciones de perdida . . . 20
    - 2.6.2.4. Métodos de optimización . . . 21
  - 2.6.3. Redes convolucionales . . . 22
    - 2.6.3.1. Capas de extracción de características . . . 25
    - 2.6.3.2. *DropOut* . . . 28
    - 2.6.3.3. Normalización por lotes . . . 28
    - 2.6.3.4. *Global Average Pooling* . . . 29
  - 2.6.4. YOLO . . . 30
- 2.7. Filtro de Kalman . . . 31
- 2.8. Tecnologías utilizadas . . . 34
  - 2.8.1. Python . . . 34

2.8.2. Docker . . . 34
2.8.3. Nvidia Docker . . . 35
2.8.4. Tensorflow . . . 36
2.8.5. Conjunto de datos COCO . . . 37
2.8.6. Autotrack . . . 37

3. Estado del Arte **38**
3.1. Sistemas basados en la carretera . . . 38
3.2. Sistemas basados sobre la carretera . . . 40
3.3. Sistemas basados a un lado de la carretera . . . 43
3.4. Comparación de trabajos desarrollados . . . 44

4. Metodología **46**
4.1. Análisis y diseño del sistema . . . 46
4.1.1. Actores . . . 47
4.1.2. Casos de uso . . . 48
4.1.3. Diagrama de contexto . . . 48
4.1.4. Arquetipos . . . 49
4.1.5. Arquitectura . . . 50
4.2. Sistema extractor de características . . . 52
4.2.1. Extracción de fotogramas . . . 52
4.2.2. Detección y clasificación . . . 53
4.2.3. Seguimiento, frecuencia y reapariciones . . . 55
4.2.4. Conteo vehicular . . . 56
4.3. Preparación de entrenamientos . . . 57
4.3.1. Preparación de imágenes . . . 57

5. Resultados **59**
5.1. Caso de estudio . . . 59
5.2. Entrenamientos de modelo clasificador . . . 60
5.2.1. Conjunto de datos . . . 60
5.2.2. Preparación de conjunto de datos . . . 61
5.2.3. Hardware utilizado . . . 64
5.3. Configuración y análisis de experimentos . . . 65

6. Conclusiones **68**
6.1. Conclusión . . . 68
6.2. Aportaciones . . . 69
6.3. Trabajos a futuro . . . 70

Referencias **71**

Índice de figuras

2.1. Representación de una red neuronal multicapa 13
2.2. Representación de funcionamiento de propagación hacia atrás 14
2.3. Relación entre los conceptos de inteligencia artificial, aprendizaje máquina y aprendizaje profundo 16
2.4. Comparación entre el aprendizaje profundo y los algoritmos tradicionales . . 17
2.5. Representación del funcionamiento de la tasa de aprendizaje 19
2.6. Representación de una simple red convolucional (Zaniolo & Marques, 2020). 23
2.7. Representación del kernel 24
2.8. Representación del Stride 25
2.9. Representación de la convolución (Kim, 2017). 26
2.10. Representación del Max-Pooling 27
2.11. Representación visual de *DropOut* (Srivastava et al., 2014b). 28
2.12. Representación *visual Global Average* Pooling 30
2.13. Representación de datos extraídos con red YOLO 31
2.14. Representación de arquitectura de red neuronal de YOLOv3 (Redmon & Farhadi, 2018). 31
2.15. Representación del ciclo que utiliza el Filtro de Kalman 32
2.16. Algoritmo de filtro Kalman (Welch, Bishop et al., 1995). 33
2.17. Representación de arquitectura de funcionamiento de Docker 35
2.18. Arquitectura de Nvidia Docker (Overview s.f.). 36
2.19. Clases incluidas en el conjunto de datos COCO (Lin et al., 2014) 37

4.1. Etapas de desarrollo 46
4.2. Diagrama de casos de uso 48
4.3. Diagrama de contexto 49
4.4. Representación de la interacción de los arquetipos 50
4.5. Representación del proceso de extracción de información vial 52
4.6. Representación de extracción de fotograma 53
4.7. Representación de matriz de fotogramas con píxeles normalizados 53
4.8. Funcionamiento de modelo predictivo 54
4.9. Detección de vehículos en punto B, sus trayectorias y triangulación del generado con el punto A y B. 55
4.10. Representación de formato de etiqueta de datos 58

5.1. Lugar donde se tomaron las muestras 60

5.2. Representación de coordenadas de cuadro delimitador 62
5.3. Interfaz de LabelImg 63
5.4. Muestra de imágenes de conjunto de datos junto a su etiqueta 64
5.5. Gráfica de resultados de entrenamientos 66

Índice de tablas

3.1. Tabla comparativa entre trabajos del estado del arte 45

4.1. Casos de uso 48

5.1. Cantidad de imágenes del conjunto de datos por clase 61
5.2. Resultados obtenidos en entrenamientos con distintos parámetros 65
5.3. Precisión de clasificación obtenida por clase 66
5.4. Comparación de resultados entre trabajos del estado del arte 67

Capítulo 1

Introducción

En la actualidad, el incremento de la población humana en las ciudades ha provocado el aumento de la cantidad de vehículos automotores en circulación, generando tráfico, caos vial e incluso aumento en accidentes. Los gobiernos, buscando alternativas para mejorar u optimizar vialidades existentes y futuras, necesitan obtener información del tráfico vial. Algunos de los datos que se necesita conocer son: clasificación de tipos de vehículos, frecuencia vehicular, entre otros. Los departamentos encargados de tomar decisiones en relación con las vialidades utilizan distintas técnicas para recopilar estos datos.

Existen en la literatura distintas tecnologías de detección, clasificación y recopilación de información vial, como las cámaras, sensores, drones y radares. En (Won, 2020) se muestra una taxonomía de los sistemas de clasificación vial, donde se agrupan los distintos tipos de sistemas en tres grupos: basados en la carretera, sobre la carretera y a un lado de la carretera.

El grupo de sistemas basados en la carretera son aquellos que utilizan dispositivos que deben ser instalados o incrustados sobre el pavimento. Aquí se utilizan sensores, fibras sensibles, detectores, entre otros dispositivos. Todos estos detectan el paso de los vehículos sobre ellos. Por otro lado, se encuentran los que hacen uso de detectores de bucle inductivos (Coifman & Neelisetty, 2014). Se trata de un circuito cerrado hecho de cable de cobre que se instala en la carretera. Este captura los cambios de inductancia y genera una señal cuando los vehículos pasan sobre él. También existen sistemas que utilizan sensores magnéticos (Balid et al., 2018). Estos se benefician de las distorsiones provocadas por el chasis de los vehículos en el campo magnético para hacer clasificación de acuerdo al tamaño de la distorsión provocada. Una característica de este tipo de sistemas es, como su nombre lo indica, que deben ser

instalados en la carretera, afectando al tráfico durante su instalación. Algo a tomar en cuenta de este tipo de sistemas es que en su mayoría deben hacer modificaciones en la carretera llegando a detener el tráfico para instalarse.

El segundo grupo que se presenta son los sistemas basados sobre la carretera. Este tipo de sistemas utilizan dispositivos en su mayoría cámaras, instaladas arriba de la carretera. Existen dos ramificaciones, los que se instalan en alguna estructura o puente sobre la carretera y los aéreos. En el primer apartado se encuentran aquellos que utilizan sobre todo cámaras para extraer características viales. En el apartado de dispositivos aéreos son aquellos que utilizan satélites y vehículos aéreos no tripulados UAV (UAV, por sus siglas en inglés) (Niu et al., 2018). Estos últimos tienen la característica de cubrir grandes áreas, pero con la desventaja de la baja resolución en sus imágenes, haciendo que la tarea de detección y clasificación no sea algo sencillo.

El último grupo son los sistemas basados a un lado de la carretera. Son aquellos donde los dispositivos se instalan a un lado de las carreteras. Los sensores magnéticos (Quan et al., 2020) se utilizan también en los sistemas basados a un lado de la carretera. Funcionan con el mismo principio de utilizar las perturbaciones generadas por los vehículos. Además, otros dispositivos que utilizan este tipo de desarrollos, son sensores acústicos (H. Liu et al., 2020), LIDAR (Lee & Coifman, 2015), RF (Kassem et al., 2012) y Wi-Fi (Won et al., 2017). La mayoría de todos estos sistemas utilizan infraestructura especializada, lo cual requiere de una gran inversión inicial. También, algo a tomar en cuenta es que algunos de estos sistemas tienen capacidad de recopilar pocos tipos de datos, limitándose a unos cuantos o inclusive solo uno.

Un supuesto importante en este trabajo es que el número de cámaras instaladas en todo el mundo ha ido creciendo en los últimos años. El parámetro más relevante para la monitorización del tráfico es la velocidad del coche, ya que cuando se utiliza con el recuento de vehículos, permite determinar el comportamiento de la movilidad motorizada en las ciudades. Por lo tanto, la estimación precisa de la velocidad de los vehículos es un problema abierto para los Sistemas Inteligentes de Transporte (ITS, por sus siglas en inglés). Cuando se consideran las cámaras para obtener la velocidad del vehículo, hay que considerar la traslación de un plano 3D a un plano discreto 2D. Esta limitación intrínseca da lugar a una representación

digital cuya precisión sigue la ley del cuadrado inverso; su cantidad es inversamente proporcional al cuadrado de la distancia de la cámara al vehículo. Cuando se utiliza una cámara monocular para obtener imágenes de vídeo con el fin de estimar la distancia de los vehículos, normalmente se considera un conjunto de restricciones, como la carretera plana, incluyendo métodos basados en la homografía y el uso de líneas, patrones o regiones de intrusión aumentadas, o conocimiento previo sobre las dimensiones reales de algunos de los objetos (por ejemplo, las placas de matrícula o el tamaño de los vehículos (Fernández Llorca et al., 2021)).

Este trabajo propone el desarrollo de un sistema que a partir de secuencias de imágenes de tráfico genere información vial, esto haciendo uso de infraestructura ya existente, como lo son las cámaras de vigilancia que se encuentran instaladas en distintos puntos de las ciudades. Gracias a esto, se evita realizar una gran inversión en dispositivos especializados como sensores, drones, radares, entre otros. El sistema propuesto cuenta con las funciones de detección, clasificación, rastreo y frecuencia vehicular, tomando en cuenta obstrucciones y reparación de vehículos.

El sistema desarrollado se encuentra en la clasificación de sistemas basados a un lado de la carretera. Las muestras, que se obtuvieron de videos de tráfico, se realizaron con un celular instalado a un lado de una vía principal. El desarrollo de este sistema es capaz de extraer clasificación, frecuencia y trayectoria vehicular, esta información se almacena en un archivo CSV, el cual se consulta una vez terminado el análisis del video. Como resultado del desarrollo de este trabajo se obtuvo un modelo clasificador de vehículos, este modelo se entrenó con un total de 21000 imágenes dentro de siete clases de vehículos. El modelo clasificador logró una precisión de hasta un de 95.43 %.

Al final se realiza una comparación con otros trabajos del estado del arte donde se observa un buen comportamiento y precisión competitiva de nuestro trabajo. Cabe destacar que otros trabajos realizan solo detección y clasificación de vehículos, en cambio, nuestro sistema extrae otras características viales mencionadas anteriormente.

1.1. Definición del problema

La información vial es de suma importancia para los departamentos de tránsito de las ciudades, con ella se toman importantes decisiones con el objetivo de ofrecer una mayor seguridad para la población, buscando reducir accidentes. Tan solo en México en el año 2020 la suma llego a los 301,678 (INEGI, 2020). Además, se busca extraer características viales como, frecuencia, clasificación y conteo vehicular. Estos datos se ofrecen a los departamentos de tránsito en busca de realizar análisis de tráfico y mejorar el flujo vehicular, ofreciendo rutas alternativas, implementando semáforos, planeando mejores estrategias en la infraestructura de carreteras, entre otras alternativas.

Actualmente, existen distintas tecnologías para la extracción de información vial, algunas de estas opciones son: sensores, radares, drones, cámaras, incluso por medio de mano de obra humana. En el caso de la mano de obra humana se limita la cantidad de información que se llega a obtener. Otros dispositivos como los sensores y radares se restringe a la cantidad de tipos de información que se llega a obtener, llegando en algunos casos a solo clasificar o hacer conteo de vehículos.

Algunos dispositivos, como los sensores o bucles inductivos, son bastante invasivos, hasta el punto de detener el tráfico para poder instalarse y en ocasiones dañando la infraestructura. Por ejemplo, los detectores de bucle inductivo son dispositivos que se instalan en la propia carretera. Lo cual hace que durante su instalación se deba detener el tráfico o generar un cuello de botella, aumentando el tiempo de circulación de las personas.

Cuando se utilizan dispositivos como los antes mencionados (sensores, radares), la clasificación de vehículos se limita en muchos casos a solo clasificar por tamaño o peso. En cambio, con el uso de cámaras, se pueden obtener mayor cantidad de características. Algunas son: distancia y tiempo de recorrido, mayor número de clases para clasificación e incluso en algunos trabajos los caracteres de la placa vehicular (Luvizon et al., 2014). Esto ofrece mayor información para hacer un análisis de tráfico.

Por otra parte, el costo de algunos dispositivos especializados, junto con el material para su instalación y puesta en marcha, llega a ser elevado debido a los materiales utilizados y la exactitud que ofrecen en los datos. Así mismo, el costo también se eleva si el sistema debe

ser instalado o supervisado por personal calificado.

Se han llevado a cabo implementaciones de sistemas de extracción de características de tráfico, utilizando visión artificial y redes neuronales. Algunos de estos trabajos tienen ciertas limitaciones, algunos en su implementación, otros en las condiciones necesarias para su funcionamiento, incluso algunos utilizan cámaras especiales para las tareas de detección y clasificación. Cabe mencionar que muchos de ellos no cuentan con el manejo de obstrucciones, lo cual implica la perdida de información de algunos vehículos.

El uso de cámaras para los sistemas de tráfico inteligente, al igual que otros dispositivos, trae consigo consideraciones a tomar en cuenta. Por ejemplo, la resolución de la cámara. No es lo mismo trabajar con imágenes de 1920x1080 píxeles a imágenes de 720x480 píxeles. La información contenida no es la misma, en uno de los casos contiene un mayor número de características. Esto provoca que se necesite un mayor poder computacional para su procesamiento. No obstante, esa no es la única consideración. Cuando se toma un video, el espacio que se captura es un área 3D; sin embargo, las imágenes pasan a un plano 2D donde las dimensiones, se encuentran distorsionadas. Por lo tanto, la calibración de la cámara en estos sistemas es algo a tomar en cuenta a la hora de hacer cálculos de las posiciones y las distancias que los vehículos recorren. Algunos trabajos optan por utilizar cámaras especiales o dos cámaras para evitar problemas de calibración. Con la desventaja de aumentar su costo de inversión por la utilización de varios dispositivos.

Propuesta de solución

Se propone el uso de distintas herramientas y algoritmos para llevar a cabo las distintas tareas de extracción de características. Para el desarrollo del sistema se utiliza Python y distintas librerías y algoritmos. Por ejemplo, OpenCV cuenta con una función para obtener fotogramas de un video. Estos fotogramas se analizan con un modelo clasificador de redes convolucionales, obteniendo los primeros datos de interés, la región de interés (ROI, por sus siglas en inglés) y la clase del objeto detectar. El ROI es el área donde el modelo predictivo detecta un vehículo. Esta área es delimitada por dos pares ordenados de números, correspondientes a dos esquinas de un rectángulo, el cual indica el área de interés. Este proceso lo realiza automáticamente el modelo clasificador de YOLOv3 (Redmon & Farhadi, 2018). Una

vez obtenida esta área, las redes convolucionales detectan las características del objeto y lo clasifican dentro de alguna de nuestras clases de vehículos o personas. Las clases implementadas para el proyecto son: auto, camioneta, autobús, van, motocicleta, bicicleta y persona. Para obtener un modelo clasificador se propone el uso de Darknet junto a las redes convolucionales de YOLO, las cuales permiten hacer entrenamientos personalizados y supervisados.

El seguimiento de vehículos y la extracción de frecuencia vehicular se realizará con la herramienta AutoTrack (Burnett et al., 2019). Esta herramienta recopila todas las posiciones de los objetos que se detectan en los fotogramas. Cada nuevo vehículo detectado se le asigna un identificador y se guarda en una lista donde se almacena junto con sus datos obtenidos. Una función dentro de la tarea de seguimiento es la detección de obstrucciones. Cuando un vehículo se oculta, detrás de otro y en algún momento vuelve a reaparecer, se detecta como un vehículo anteriormente detectado y no se vuelve a contar.

Para que estos algoritmos trabajen en conjunto se propone el diseño de una arquitectura la cual los conjugue y permita el envío de información entre los distintos procesos. De esta forma, el sistema iniciara el procesamiento de un video y lograra extraer las características de interés.

1.2. Hipótesis

Extraer características de tráfico vial y vehículos con técnicas de visión artificial y aprendizaje profundo permitirá obtener datos de clasificación y frecuencia vehicular en situaciones no controladas.

1.3. Objetivo

Desarrollar un sistema extractor de información de tráfico vial como detección, clasificación y frecuencia vehicular, además de detectar reaparición de vehículos después de una obstrucción, a partir de una secuencia de tráfico vial en situaciones no controladas.

Objetivos específicos

- Construir un conjunto de datos de vehículos de las siguientes clases: auto, camioneta, autobús, van, motocicleta, bicicleta y persona.
- Seleccionar e implementar un algoritmo de redes convolucionales, que permita realizar detección y clasificación vehicular a partir de secuencia de imágenes.
- Desarrollar e implementar un algoritmo que permita obtener frecuencia vehicular tomando en cuenta obstrucciones.
- Seleccionar e implementar de un algoritmo que permita realizar seguimiento de múltiples.
- Diseñar e implementar una arquitectura que permita utilizar en conjunto los algoritmos de detección, clasificación, frecuencia y seguimiento vehicular.
- Validar y analizar resultados obtenidos del sistema extractor de información vial desarrollado.

1.4. Justificación

El uso de herramientas de visión artificial y redes neuronales entrenadas para obtener información vial a partir de secuencias de video, ofrece a las instituciones interesadas una gran fuente de datos necesarios para llevar a cabo un análisis vial en vista de tomar decisiones que mejoren el flujo vehicular y evitar congestión de autos en las calles, así como disminuir accidentes que pongan en riesgo vidas humanas.

El uso de sistemas de información vial ofrece a las instituciones interesadas una gran ventaja a la hora de obtener datos del tráfico vial. Estos facilitan también la toma de decisiones al momento de hacer una planeación de rutas alternativas o futuras autopistas para mejorar el flujo vehicular. Esto tiene impacto en aspectos como el tiempo en que los ciudadanos utilizan los vehículos, reduciendo el nivel de contaminación generada por la quema de combustible. Además de una reducción de accidente provocada por la congestión de autos.

El sistema está diseñado para utilizar la infraestructura existente como lo son las cámaras de videovigilancia. Estas se encuentran ya instaladas en las vialidades en diversas ciudades. Por lo que no es necesario realizar instalaciones costosas o invertir en equipo especializado, así como en personal especializado para su operación. Lo que conlleva a una reducción significativa en costos. También el equipo de cómputo no requiere una gran inversión. Lo anterior debido a que el diseño de la aplicación permite ser utilizada por equipos que no cuenten con un alto poder computacional. El sistema es adaptable a trabajar tanto con una Unidad Central de Procesamiento (CPU, por sus silgas en inglés) como es una Unidad Gráfica de Procesamiento (GPU, por sus siglas en inglés).

Los dispositivos especializados como lo pueden ser los radares de velocidad, los sensores magnéticos, las mangueras de inducción, entre otros, son capaces de obtener pocos tipos de datos. Algunos de ellos solamente detectan el paso de vehículos. Por ejemplo, los sensores magnéticos detectan los cambios en el campo magnético generados por los autos cuando circulan por encima de ellos. Es decir, solo detectan el paso de los vehículos. En cambio, el uso de técnicas de aprendizaje profundo y visión artificial ofrece una amplia mejora en comparación a estos dispositivos. Debido a que es una imagen la que se analiza, se pueden extraer distintas características como: clasificación exacta, detección, tiempos de recorrido, distancias recorridas, velocidad del vehículo, conteo exacto e incluso hacer un seguimiento del vehículo. Estos datos ofrecen mayor información para realizar un análisis más detallado a las instituciones interesadas. La capacidad de las calles es limitada. Por lo que se debe analizar constantemente la frecuencia de autos para optar por vías alternativas o mejorar la infraestructura existente. Además, este dato, junto a la clasificación, ofrecen información cuantitativa para los departamentos de tránsito para llevar a cabo planeaciones de mantenimiento preventivo y evitar problemas a la población como daños a sus vehículos y mejorar el flujo vehicular.

1.5. Estructura de la tesis

Este trabajo está dividido en seis capítulos organizados de la siguiente manera:

- Capítulo 1 - Introducción: Se define el problema, la hipótesis, el objetivo, los objetivos

específicos y la justificación de este trabajo.

- Capítulo 2 - Marco teórico: Se presentan los conceptos clave de la inteligencia artificial, tales como visión artificial, aprendizaje máquina, aprendizaje profundo, redes neuronales, además de las tecnologías utilizadas que forman parte del desarrollo de este proyecto.

- Capítulo 3 - Estado del arte: Se presentan diversos trabajos que se encuentran en el estado del arte que abordan la extracción de características de tráfico vial, dando un resumen de su propuesta y resultados, así como haciendo una comparación contra el trabajo aquí presentado.

- Capítulo 4 - Metodología: Se muestra el proceso que se llevó a cabo para desarrollar este proyecto, así como la explicación de las decisiones que se tomaron para el diseño de la arquitectura y las funciones.

- Capítulo 5 - Análisis de los resultados: Se presentan los resultados que se generaron del funcionamiento de la aplicación, un análisis detallado tomando en cuenta parámetros de evaluación, así como comparación con otras herramientas.

- Capítulo 6 - Conclusión: Se presenta un resumen de la estructura del proyecto, así como el aporte que se lleva a cabo a raíz del desarrollo de esta aplicación y propuestas para trabajos a futuro con base en los resultados obtenidos en este proyecto.

Capítulo 2

Marco Teórico

En este capítulo se mostrarán temas relacionados con el desarrollo de este trabajo, los cuales son útiles para comprender todos los temas y técnicas utilizadas en cada una de las secciones por mostrar. La mayoría de estos temas están relacionados con la inteligencia artificial y el aprendizaje profundo. En un inicio se explica la historia de la inteligencia artificial y como con el paso de los años ha ido desarrollándose hasta tomar un papel muy importante en la actualidad.

2.1. Inteligencia Artificial

La inteligencia se describe como la capacidad de realizar tareas, de entender y/o comprender la realidad y adaptarse a ella. Teniendo este concepto en mente, en computación la rama de la inteligencia artificial se encarga de producir máquinas inteligentes que cuenten con esta capacidad.

Historia

En el año de 1943 Warren McCulloch y Walter Pitts fueron los primeros en hacer un trabajo reconocido en el campo de la inteligencia artificial, ellos propusieron un modelo de redes neuronales donde cada neurona tenía dos posibles estados, encendido o apagado (Mcculloch & Pitts, 1943). Con esto demostraron que se podría calcular cualquier función computable con el uso de redes neuronales. Sin embargo, con los siguientes experimentos se encontró que no se podía representar el estado de una neurona solamente de manera binaria, sino que

había otras características que se debían tomar en cuenta. Gracias a esto y al poco poder de procesamiento que se tenía en ese entonces, durante años los avances en el campo de la inteligencia artificial fueron lentos.

En 1958 Frank Rosenblatt, presento el primer algoritmo funcional basado en el perceptrón, el cual era una generación de la neurona de Mculloch y Pitts, este podría "aprender" a través de ponderación para cada entrada y su funcionaba como un clasificador binario. Lamentablemente en 1969, luego de tener una sobre expectación de los resultados de este campo, Marvin Minsky y Seymour publicaron resultados poco favorables para la inteligencia artificial, compartiendo las limitaciones que se tenían hasta ese momento (Minsky & Papert, 1969). Esto trajo un "invierno" en el campo de la inteligencia artificial.

A comienzos de la década de 1980 se tuvo un gran avance de la mano de Geoffrey Hinton y algunos colegas, los cuales dieron a conocer un método llamado Backpropagation, el cual permitía que una red neuronal aprendiera a partir de una muestra de datos. Esto animo a los investigadores e inversionistas y creo un nuevo resurgimiento en el área del aprendizaje profundo.

2.2. Aprendizaje máquina

El aprendizaje máquina consta de utilizar algoritmos que puedan extraer características de datos en bruto y representarla en algún tipo de modelo. Este modelo se utiliza para inferir cosas sobre otros datos que aún no se han modelado.

Las redes neuronales son un tipo de aprendizaje máquina, que existen desde hace al menos 50 años. La unidad más pequeña de estas redes se basa en la neurona del cerebro animal, así como sus conexiones también se basan en el cerebro de los mamíferos. El aprendizaje máquina es una rama de los algoritmos computaciones que están diseñados para emular la inteligencia humana aprendiendo del entorno.

Las técnicas del aprendizaje máquina se han utilizado con éxito en diversos campos que van desde el reconocimiento de patrones, la visión por ordenador, la ingeniería de naves espaciales, las finanzas, el entretenimiento y la biología computacional hasta aplicaciones biomédicas y médicas.

Un algoritmo de aprendizaje automático es un proceso computacional que utiliza datos de entrada para lograr una tarea deseada sin estar literalmente programado para producir un resultado concreto. Estos están codificados de forma suave, ya que se adaptan automáticamente su arquitectura a través de la repetición para ser cada vez mejor para la tarea deseada.

El proceso de adaptación se llama entrenamiento en el que se proporcionan muestras de datos de entrada junto con resultados deseados. Este entrenamiento es la parte de aprendizaje. Este algoritmo sigue aprendiendo nuevas características incluso después del entrenamiento a medida que vaya procesando nuevos datos y aprendiendo de sus errores. Existen tres tipos de aprendizaje automático (Patterson & Gibson, 2017):

- **Supervisado**. - en este entrenamiento cada ejemplo del conjunto de datos de entrada del entrenamiento se empareja con una etiqueta, la cual especifica la clasificación del dato. De esta forma el modelo comienza a conocer características únicas de cada clase del entrenamiento y es más sencillo llevar a cabo una clasificación precisa.

- **No supervisado**. - se trata de un entrenamiento no supervisado debido a que no se asocia una determinada etiqueta a los datos de entrada. De modo que el algoritmo aprende por si solo las características de cada clase involucrada solamente con la lectura y procesamiento de los datos.

- **semi-supervisado**. - este algoritmo utiliza una parte de datos sin etiquetar y otra parte de datos etiquetados, esto es útil cuando una parte de los datos no se encuentra etiquetada y el entrenamiento se apoya de la parte de datos etiquetados para aprender características que también se encuentren en los datos sin etiquetar.

2.3. Redes Neuronales

Las redes neuronales fueron creadas con el propósito de imitar la forma en que el cerebro animal aprende, por ello comparten algunas propiedades. La forma en que la red va a trabajar está dictaminada por su arquitectura, en esta se definen: el número de neuronas, número de capas, tipos de conexiones entre capas, función de activación, etc.

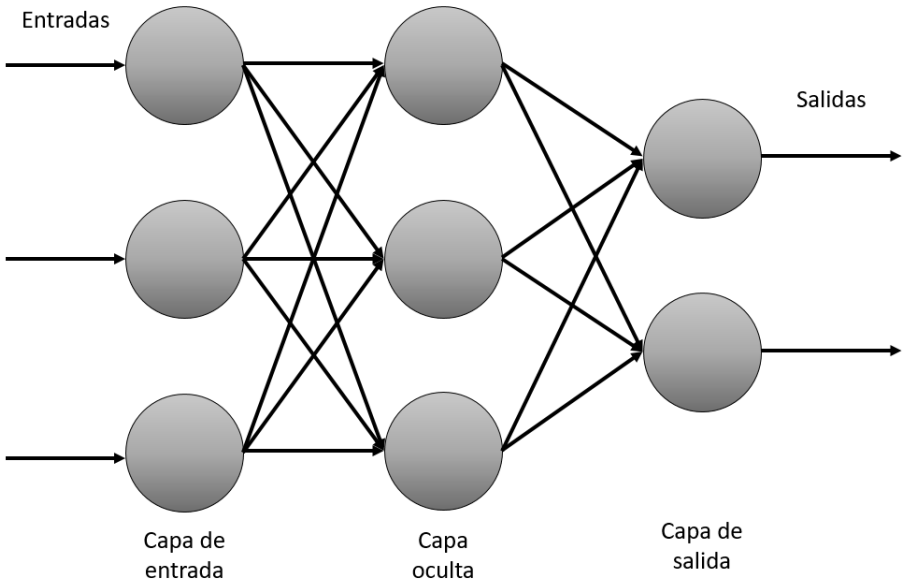

Figura 2.1: Representación de una red neuronal multicapa

Para que la red pueda aprender de los datos, se le asignan pesos sinápticos a cada neurona, los cuales se deberán ir actualizando para lograr disminuir su error en los resultados. La función de activación es la responsable de ir obteniendo nuevos pesos para poder actualizarlos. La actualización de los pesos es el corazón de las redes neuronales artificiales, es el medio de almacenamiento de información y es la manera en que la red va aprendiendo nueva información.

2.3.1. Redes multicapa

La red neuronal más conocida y fácil de entender es la red neuronal multicapa. Tiene una capa de entrada, una o más capas ocultas y una única capa de salida. Es posible definir cada capa con un número diferente de neuronas, y cada capa está completamente conectada con la capa anterior (Swamynathan, 2017).

2.3.2. Propagación hacia atrás

El algoritmo de Propagación hacia atrás o retro propagación se introdujo originalmente en la década de 1970, pero su importancia no se apreció por completo hasta después de un fa-

moso artículo de 1986 de David Rumelhart, Geoffrey Hinton y Ronald Williams (Rumelhart & McClelland, 1986). Este algoritmo se compone de dos etapas, la etapa de propagación hacia adelante y la etapa de proporción hacia atrás. La arquitectura general de la red se presenta en la Figura 2.2, donde se aprecia una red neuronal de tres capas.

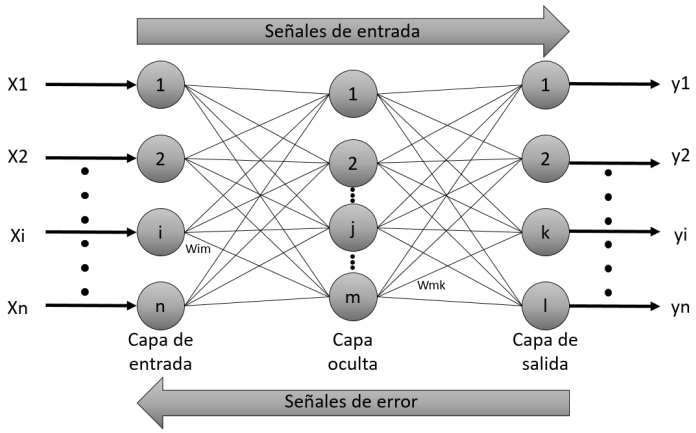

Figura 2.2: Representación de funcionamiento de propagación hacia atrás

2.4. Visión artificial

Una de las bases fundamentales para desarrollar todas las capacidades de la inteligencia artificial es permitir que las máquinas vean. Para imitar la visión humana, las máquinas deben adquirir, procesar, analizar y comprender imágenes. El aprendizaje interactivo que se realiza con la red neuronal ha hecho posible un gran progreso para lograr este paso. En el mundo de la visión artificial, el aprendizaje interactivo permite a las máquinas determinar si algo es o no una flor, una casa o una muñeca. Asimismo, estos sistemas permiten que las máquinas aprendan a reconocer imágenes de cualquier cosa, sin tener que ser programadas previamente. Para hacerlo, las computadoras emplean redes neuronales. Así, las máquinas pueden aprender a reconocer imágenes de forma similar a como los humanos aprenden a reconocer cosas en su mundo. La visión artificial ha progresado rápidamente desde que la aplicación de redes neuronales comenzó en la década de 1990. (Szeliski, 2010)

2.5. Aprendizaje por transferencia (*Transfer Learning*)

En el campo del aprendizaje máquina, a diferencia de métodos tradicionales, el aprendizaje por transferencia proporciona un método para utilizar los conocimientos adquiridos previamente y reutilizarlos para aprender nuevos conocimientos. Además, el aprendizaje por transferencia toma en cuenta que los datos de entrenamiento pueden estar dentro de otro dominio que los datos de prueba. Es decir, los datos de entrenamiento pueden ser de unos objetos en concreto y los datos de prueba ser objetos completamente diferentes. Los métodos tradicionales de aprendizaje automático realizan una predicción sobre datos que no se conocían utilizando modelos matemáticos previamente entrenados con datos etiquetados o sin etiquetar, que son del mismo tipo que los datos a los que se aplicara la predicción. En cambio, en el aprendizaje por transferencia permite que datos que se utilicen tanto en las pruebas como en el entrenamiento no deban ser del mismo tipo necesariamente. Esta técnica ha sido desarrollada tomando en cuenta la capacidad que tienen los humanos para utilizar su conocimiento previamente adquirido y resolver problemas no antes vistos, pero similares, de una manera más eficaz y rápida. Por ejemplo, el hecho de conocer lo que es una manzana, ayuda a poder reconocer una pera (Tan et al., 2018).

2.6. Aprendizaje profundo

Dentro del campo de la inteligencia artificial existe el aprendizaje máquina, y dentro de este se encuentra el aprendizaje profundo, lo que hace que este último sea una especialidad de la inteligencia artificial y se puedan utilizar técnicas propias de inteligencia artificial y de aprendizaje máquina. En la Figura 2.3 se muestra una representación visual de la relación entre estos tres conceptos.

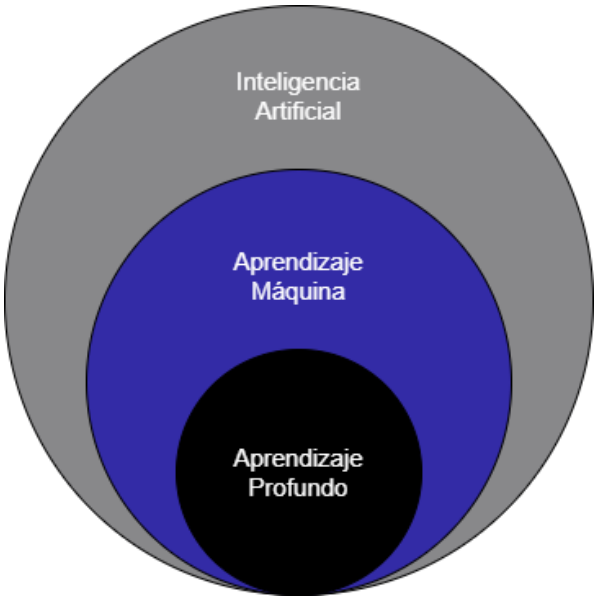

Figura 2.3: Relación entre los conceptos de inteligencia artificial, aprendizaje máquina y aprendizaje profundo

Actualmente, no existe una definición propia o exacta de aprendizaje profundo debido a que desde sus inicios existe una constante evolución. No obstante, se acepta la definición "Red neuronal con más de dos capas", más, sin embargo, hoy en día es probable que esta definición esté desactualizada debido al enorme tamaño que algunas redes poseen, y más al saber que existen redes neuronales de más de dos capas desde hace más de cuatro décadas. Algunas características extras con las que podemos identificar el aprendizaje profundo son las siguientes:

- Modelos con una gran cantidad de neuronas
- Conexiones más complejas entre las capas y las neuronas
- Necesidad de un mayor poder computacional para la creación de los modelos
- Extracción de características automático

El término, "una gran cantidad de neuronas", se refiere a la evolución en la arquitectura de las redes neuronales, aumentando significativamente el tamaño de las capas, el número de neuronas y la complejidad de los modelos a lo largo de los años. Cuando se habla de conexiones más complejas, se refiere a que se utilizan más parámetros para optimizar modelos

resultantes, por esa razón es necesario un poder computacional más alto al que se requería incluso hace algunos años. Por último, pero incluso una de las principales características del aprendizaje profundo es la extracción automática de características. En comparación de redes tradicionales, las redes profundas son capaces de obtener una mayor cantidad y mejor calidad de las características extraídas de los conjuntos de datos, por ese motivo es posible construir modelos en una mayor número de espacios de problema, por ejemplo reconocimiento de imágenes, reconocimiento de patrones, clasificación de objetos, predicción de datos, entre otras. En la Figura 2.4 se muestra una comparación gráfica entre los algoritmos de aprendizaje máquina comunes y los de aprendizaje profundo, donde se nota la evolución en el desempeño, así como el tamaño de datos que estos últimos pueden manejar (Patterson & Gibson, 2017).

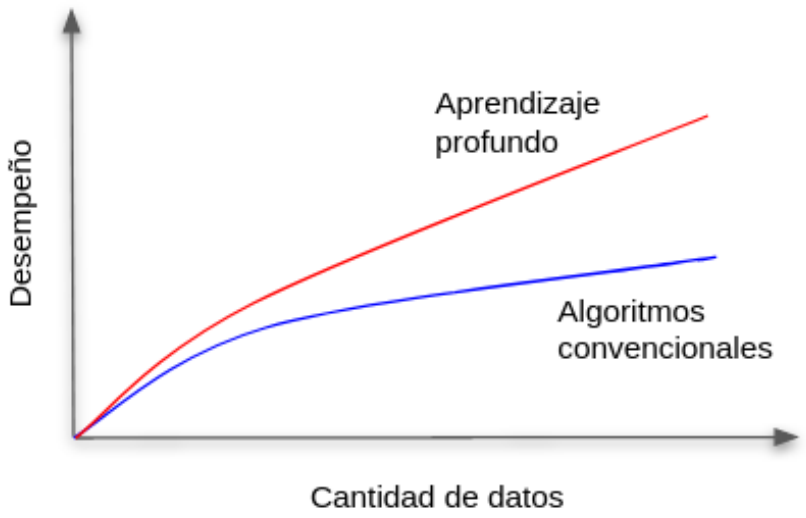

Figura 2.4: Comparación entre el aprendizaje profundo y los algoritmos tradicionales

Algunos ejemplos de redes profundas que existen son las siguientes:

- **Redes pre entrenadas sin supervisión**.- este tipo de redes cuentan con algunas características aprendidas, y con base en los datos que se sigan ingresando va adquiriendo mayor conocimiento.
- **Redes Neuronales Convolucionales**.- este tipo de redes basan su funcionamiento en las convoluciones, que son filtros extractores de características de las imágenes, por

esta razón son utilizadas mayormente para tareas de clasificación.

- **Redes Neuronales Recurrentes**.- en este tipo de redes el resultado de una neurona depende del resultado de las neuronas anteriores e incluso de la siguiente.

2.6.1. Arquitectura de redes profundas

Para poder entender aún más las redes neuronales profundas, es necesario conocer los componentes centrales que definen su funcionamiento, los cuales se muestran a continuación.

2.6.2. Hiperparámetros

El término hiperparámetros se refiere a todas aquellas variables que se configuran o ajustan al modelo de entrenamiento antes de comenzar a entrenar. Estos determinan la arquitectura, el tamaño, la capacidad de aprendizaje, por lo que influyen de manera directa a que el modelo se adapte correctamente o incorrectamente a los datos. A partir de una configuración de hiperparámetros dependerá si el modelo, tendrá sobre ajuste, sub ajuste o se adaptara correctamente al problema y obtenga buenas resultados. Para cada red neuronal y modelo de aprendizaje, los hiperparámetros a configurar cambian, a continuación, los más comunes: número de capas, número de neuronas de cada capa, tasa de aprendizaje, funciones de activación, regularización, estrategia de inicialización de pesos, funciones de pérdida, épocas de entrenamiento, algoritmos de optimización, etc. (Patterson & Gibson, 2017).

2.6.2.1. Tasa de aprendizaje

Este es un valor que se define antes del entrenamiento, es el encargado de controlar el cambio de los pesos entre las neuronas en cada iteración que se realice. Este valor debe estar balanceado, debido a que un cambio muy pequeño puede provocar que la red quede atrapada en un mínimo local o que se tarde mucho tiempo en llegar al mínimo global, por otra parte, con una tasa muy alta los resultados podrían pasarse del mínimo global, y aunque el aprendizaje sea más rápido este no estará optimizado (Rosebrock, 2017).

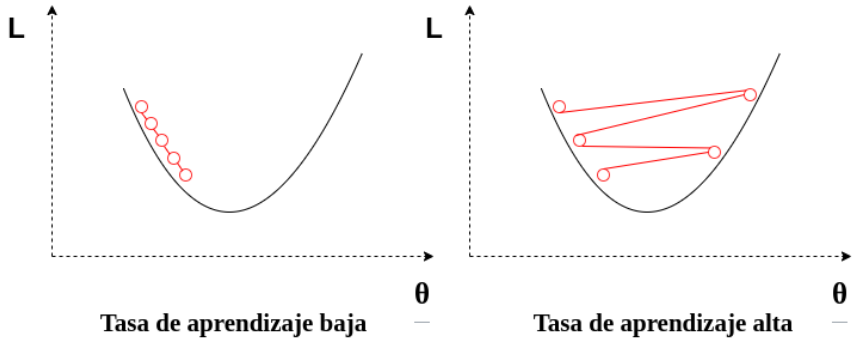

Figura 2.5: Representación del funcionamiento de la tasa de aprendizaje

2.6.2.2. Funciones de activación

Tanto en las redes neuronales artificiales como en las biológicas, las neuronas no transmiten la información de entrada a la siguiente neurona tal cual como la reciben, sino que realizan un paso adicional, la función de activación, la cual se encarga de transformar la información y agregar la no linealidad al modelo resultante (Patterson & Gibson, 2017).

- ***Softmax*** La función *Softmax* es una generalización de la regresión logística que permite su aplicación en modelos de múltiples clases. Al predecir el rango de una muestra, el modelo calcula una puntuación para cada rango y pasa el vector de puntuación a través de la función *Softmax*. La Ecuación 2.1 es la representación de esta ecuación.

$$\sigma(\overrightarrow{z}) = \frac{e^{zi}}{\sum_{j=1}^{K} e^{zj}} \tag{2.1}$$

- ***Sigmoide*** Históricamente, la función *sigmoide* es la función más antigua y popular. Como resultados nos dará un valor en el rango de [0,1], la Ecuación 2.2 es su representación.

$$s(x) = \frac{1}{e+^{-x}} \tag{2.2}$$

- ***ReLu*** La función *ReLu* (Rectificador lineal) permite el paso de todos los valores positivos sin cambiarlos; sin embargo, asigna todos los valores negativos a 0, dejando todos

los valores en el rango de $[0, \infty]$. Esto se representa con la Ecuación 2.3

$$relu(x) = max(0, x) \tag{2.3}$$

2.6.2.3. Funciones de perdida

Una función de perdida indica la diferencia que existe entre el resultado de salida que la red neuronal cálculo y el resultado deseado, entre menor sea el resultado obtenido, más eficiente es la red. Estas funciones se utilizan para evaluar el desempeño que se obtiene en el entrenamiento (Goldberg, 2017).

Debido a que existen problemas de distintos tipos, también existen distintas funciones. Para un problema de regresión el objetivo es obtener un valor lo más cercano a cero, en cambio, para un problema de clasificación se requiere que el valor sea lo más cercano a uno para una de las clases para la cual la red neuronal fue entrenada.

Algunas de las funciones de perdida para problemas de regresión son las siguientes:

- ***Error Cuadrático Medio (Mean Squared Error Loss, MSE, por sus siglas en inglés)***.- Se mide como la diferencia media cuadrática entre las expectativas y las observaciones reales. Solo le interesa el tamaño promedio del error, independientemente de su dirección. Sin embargo, debido a la elevación al cuadrado, las predicciones que están demasiado alejadas del valor real serán castigadas más severamente que las predicciones menos sesgadas. Además, podemos decir que MSE tiene propiedades matemáticas, que facilitan mucho el cálculo de pendientes. La Ecuación 2.4 es su representación.

$$MSE = \frac{\sum_{j=1}^{n}(y_j - y_p^j)^2}{n} \tag{2.4}$$

- ***Erros Medio Absoluto (Mean Absolute Error Loss, MAE, por sus siglas en inglés)***.- Se mide como un promedio de la suma de las diferencias absolutas entre las predicciones y las observaciones reales. Al igual que MSE, también mide la magnitud del error independientemente de su dirección. A diferencia de MSE (error cuadrático medio), MAE (error absoluto medio) requiere herramientas más complejas como la programación lineal para calcular pendientes. Además, MAE es más robusto para valores

atípicos, porque no usa cuadrados. En la Ecuación 2.5 está representada esta función de perdida.

$$MAE = \frac{\sum_{j=1}^{n} \left| y_j - y_p^j \right|}{n} \tag{2.5}$$

Mientras que para problemas de clasificación algunas de las funciones de perdida son las siguientes:

- **Bisagra**.- cuando se trata de redes para clasificación estrictas a extremos, es decir, tomar decisiones de 0 o 1, la función de perdida de bisagra es la más comúnmente utilizada para la optimización de este tipo de redes. Este tipo de perdida también suele utilizarse en modelos llamados "modelos de clasificación de margen máximo. La Ecuación 2.6 describe como la perdida de bisagra toma la decisión para la clasificación.

$$SVMLoss = \sum_{j \neq y_k} (0, s_j - s_{yk} + 1) \tag{2.6}$$

- **Logística**.- la función de perdida logística existe para aquellos casos donde las probabilidades son el punto de interés, más allá de predecir un 0 o un 1, la idea de la función de perdida logística es obtener la mayor tasa de probabilidad de acertar en la clase correcta para cada predicción hecha.

2.6.2.4. Métodos de optimización

El entrenamiento de un modelo de red neuronal implica buscar un conjunto de valores óptimos para su vector de parámetros. Se puede ver el entrenamiento como un problema de optimización de la función de perdida respecto al vector de parámetros de nuestro modelo. El algoritmo de Descenso de Gradiente Estocástico (SGD, por sus siglas en inglés) y sus variantes son probablemente los más usados para la optimización de modelos de aprendizaje, máquina y aprendizaje profundo. En su implementación más básica, SGD utiliza una tasa de aprendizaje fija, pero existen mejoras como esquemas de reducción gradual de tasa de aprendizaje o el algoritmo Momentum, con los cuales se puede acelerar el entrenamiento. Existen variantes de SGD que funcionan bajo la premisa de que los parámetros del modelo tienen

diferentes niveles de importancia en el resultado y pueden adaptar una tasa de aprendizaje diferente a cada parámetro, se les llama Algoritmos Adaptativos. Los optimizadores son un área de investigación muy activa dentro del aprendizaje máquina, algunos de los algoritmos adaptativos más populares actualmente son *AdaGRad*, *RMSProp*, *AdaDelta* y *Adam*:

- ***Adam***.- combina las ventajas de *AdaGrad* y *RMSProp*. El parámetro de entrenamiento se mantiene para cada parámetro y, además de calcular el *RMSProp*, cada parámetro de entrenamiento también se ve afectado por el gradiente de momento promedio (Kingma & Ba, 2014).

Cabe aclarar que no existe un algoritmo de optimización que sea mejor que los demás en todos los casos, esto depende en gran medida de las características del modelo y los datos con los que se entrena.

2.6.3. Redes convolucionales

Las redes convolucionales (CNN por sus siglas en inglés) son un tipo de redes neuronales construidas con el objetivo de aprender características de orden superior en los datos mediante convoluciones. Tienen un gran desempeño en el reconocimiento de objetos en imágenes, siendo las más utilizadas en las competencias de clasificación de imágenes.

Estas redes contienen varias capas ocultas, las cuales se encargan de detectar características de las imágenes como: curvas, bordes, líneas. De esta forma se van especializando hasta poder reconocer formas complejas como rostros, siluetas, animales, etc.

Para poder procesar las imágenes, estas deberán tener el tamaño adecuado para la arquitectura de la red, es decir, la red toma como entrada los píxeles de una imagen, si esta tiene un tamaño de 28 x 28 píxeles de alto y ancho, esto equivale a 784 neuronas necesarias tomando en cuenta que la imagen se encuentre en escala de grises. Si la imagen tuviera color, tendríamos una imagen en el espacio de color RGB, lo cual aumentaría el tamaño de la imagen a 28x28x3, para procesar esta imagen necesitamos 2352 neuronas en nuestra capa de entrada (Patterson & Gibson, 2017).

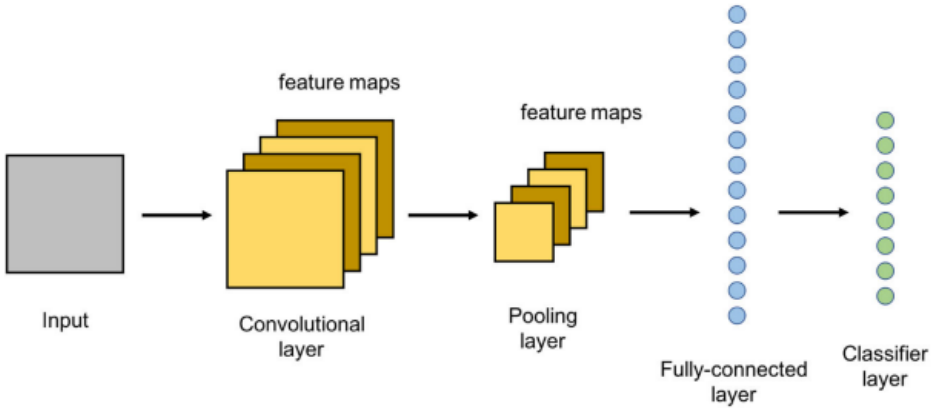

Figura 2.6: Representación de una simple red convolucional (Zaniolo & Marques, 2020).

Kernel

El kernel en las redes convolucionales se considera como el filtro que se aplica a una imagen para extraer ciertas características importantes o patrones, consiste en una "ventana", que puede ser de distintos tamaños 2x2, 3x3, 4x4, etc. Esta va recorriendo los píxeles de la imagen que se está procesando, al ir moviéndose por la imagen va haciendo una serie de cálculos que hacen que la imagen vaya reduciendo su tamaño y vayan cambiando sus valores dependiendo del tipo de filtro que se aplique. Esto hace que mientras se vaya realizando este proceso, solamente se vayan guardando las características de mayor interés para nuestra aplicación. Algunas características importantes que detecta son bordes, enfoque, desenfoque, líneas, entre otros. Normalmente, el kernel es de un tamaño menor que la imagen (Patterson & Gibson, 2017).

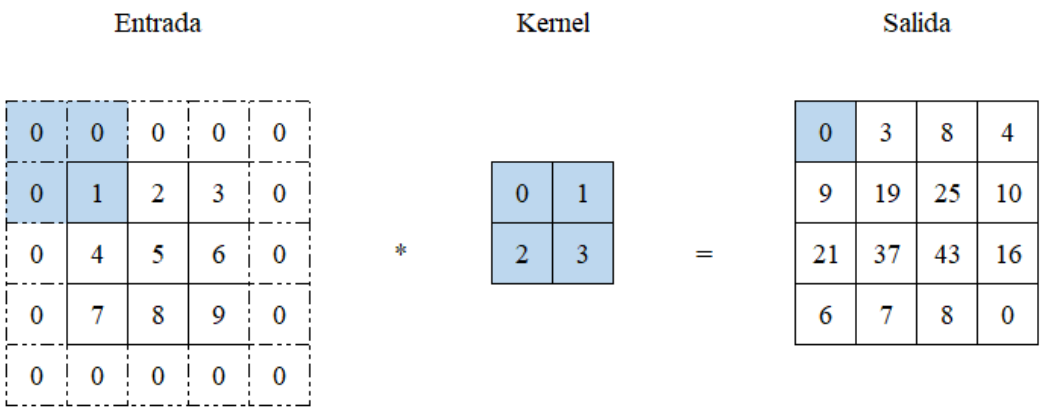

Figura 2.7: Representación del kernel

Padding

En las redes convolucionales existes distintas técnicas de preparación de datos que se realizan con el objetivo de obtener mejores resultados en la extracción de características. Una de ellas es el Padding el cual consiste en agregar píxeles de valor cero alrededor de una imagen ya representada en forma de matriz. En la Figura 2.7 podemos observar esto en la representación de la entrada, se muestra una matriz de 5x5; sin embargo, originalmente esta matriz tenía un tamaño de 3x3 y a esta se le añadieron los píxeles extra. Esta técnica se utiliza para perder la menor cantidad de información y las características que se extraen sean de mayor calidad para el modelo resultante. En la Figura 2.8, podemos observar el funcionamiento de esta técnica (Patterson & Gibson, 2017).

Stride

Para poder llevar a cabo el proceso de convolución a las imágenes, se debe aplicar un filtro a las imágenes, este filtro también llamado Kernel es el encargado de recorrer la imagen en ventanas que pueden ser de distintos tamaños 3x3, 5x5, 7x7, dependiendo de la arquitectura de la red. Pues bien, se le conoce Stride al paso que debe dar el Kernel al moverse entre la imagen, es decir, después de terminar con la primera ventada de píxeles del tamaño correspondiente, cuantos píxeles debe moverse para aplicar de nuevo el filtrado. En la Figura 2.8 se observa visualmente este paso que hace el Kernel en las imágenes (Patterson & Gibson,

2017).

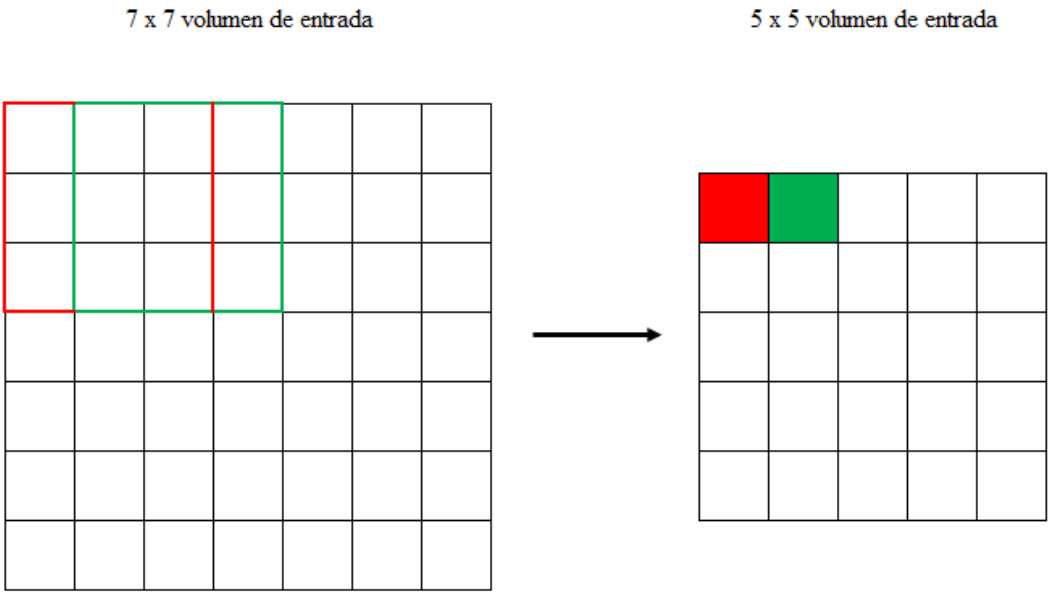

Figura 2.8: Representación del Stride

2.6.3.1. Capas de extracción de características

Las redes neuronales convolucionales repiten un patrón en su arquitectura, y es la incorporación de dos tipos de capas que se utilizan para extraer características de las imágenes, transformando el tamaño de las imágenes, haciéndolas más pequeñas y aplicando filtros. Estas dos tipos de capas son:

- Capas de convolución
- Capas de Pooling
- Capas de clasificación

Capas de Convolución

Una de las partes más distintivas en la arquitectura de las redes convolucionales se presenta en este tipo de capas de convolución o también llamadas mapas de características. La función de estas capas es tomar los datos de entrada, e ir realizando un producto escalar con

un kernel entre la región de las neuronas de la capa de entrada y los pesos a los que se encuentran conectadas localmente en la capa de salida. El kernel recorrerá todas las neuronas de entrada y obtendremos una salida con dimensiones espaciales similares (o dimensiones espaciales más pequeñas), pero en ocasiones se presenta una salida con más elementos en una tercera dimensión (dimensión de profundidad). Esto permite ir comprimiendo la imagen obteniendo solo las características más importantes de esta (Ma & J. Lu, 2017).

La Figura 2.9 muestra una representación del proceso antes descrito donde se visualiza de una manera gráfica como se aplica este filtro a la imagen de entrada y como se va calculando el producto escalar entre el filtro y los datos de entrada en los que se superpone el filtro.

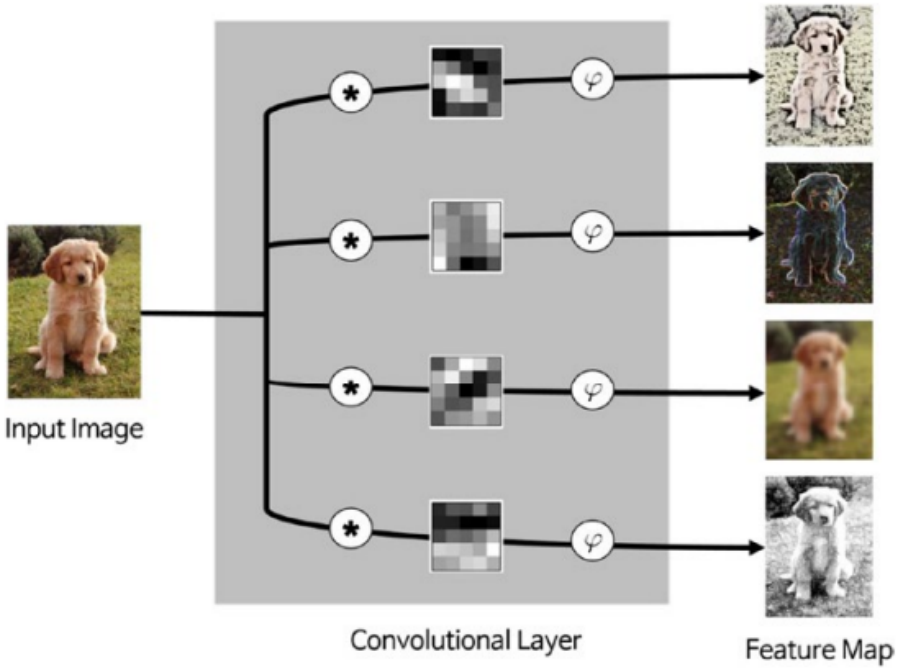

Figura 2.9: Representación de la convolución (Kim, 2017).

Capas de *Pooling*

Esta es un tipo de capa que generalmente se coloca después de una capa de convolución. Se utiliza para reducir las dimensiones de entrada para la próxima capa de convolución. Esto provoca que la información que llega a la capa de convolución siguiente sea menor, reduciendo así los cálculos que se realizan en las siguientes capas, además de reducir el sobre ajuste,

reduciendo la representación de datos en la red. Cabe señalar que estas capas son las que encuentran características en las imágenes y van construyendo progresivamente características de un orden superior.

En la siguiente figura se muestra la forma en que esta operación hace la reducción de información.

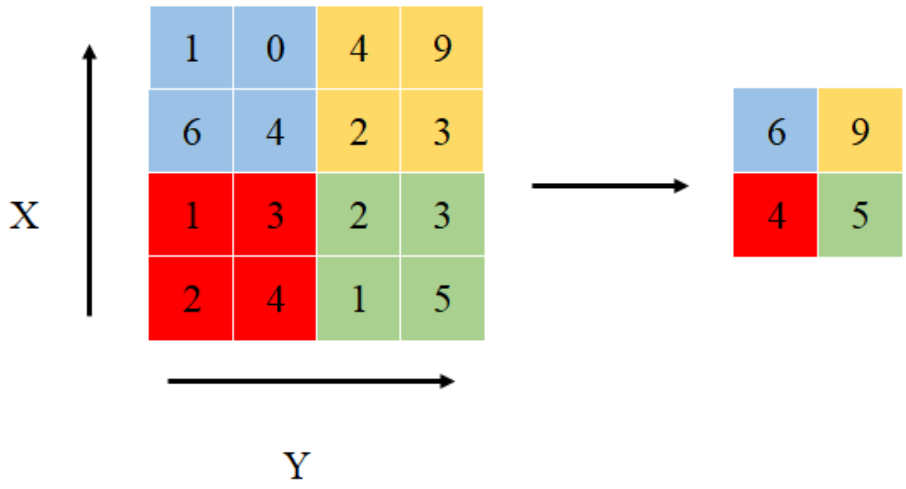

Figura 2.10: Representación del Max-Pooling

Capas de clasificación o *FullyConected*

Como su nombre lo dice, este tipo de capas, las cuales pueden ser 1 o más, se encuentran totalmente conectadas y en ellas se produce el resultado de las probabilidades o puntajes de clase, o clasificación, esto nos da a entender que es la última capa de la red neuronal. Todas y cada una de las neuronas de esta capa se encuentra conectadas a las neuronas de la capa anterior. Por lo general, la salida de esta capa es de dos dimensiones $[bxn]$, donde b es el número de ejemplos en el lote de salida y n el número a calificar.

En la Figura 2.6 se observan las distintas capas descritas trabajando juntas para llevar a cabo la tarea de clasificación, desde la capa de entrada, pasando por las capas encargadas de extraer características, convolución y Pooling, y al final pasando por la capa de clasificación y obteniendo un resultado de clasificación (Ma & J. Lu, 2017).

2.6.3.2. *DropOut*

El *DropOut* es una técnica utilizada en redes neuronales profundas, el cual tiene como objetivo evitar el sobreentrenamiento y optimizar el aprendizaje de la red. La idea es hacer una eliminación aleatoria de neuronas junto con sus conexiones a otras, durante el entrenamiento, de modo que se reduzca la sobre adaptación de la red neuronal.

Se ha demostrado que aplicar esta técnica reduce considerablemente el sobre ajuste en redes neuronales supervisadas en aplicaciones como: visión artificial de imágenes, reconocimientos del habla, clasificación de documentos y biología computacional (Srivastava et al., 2014a).

La Figura 2.11 muestra una representación visual del funcionamiento de *DropOut*.

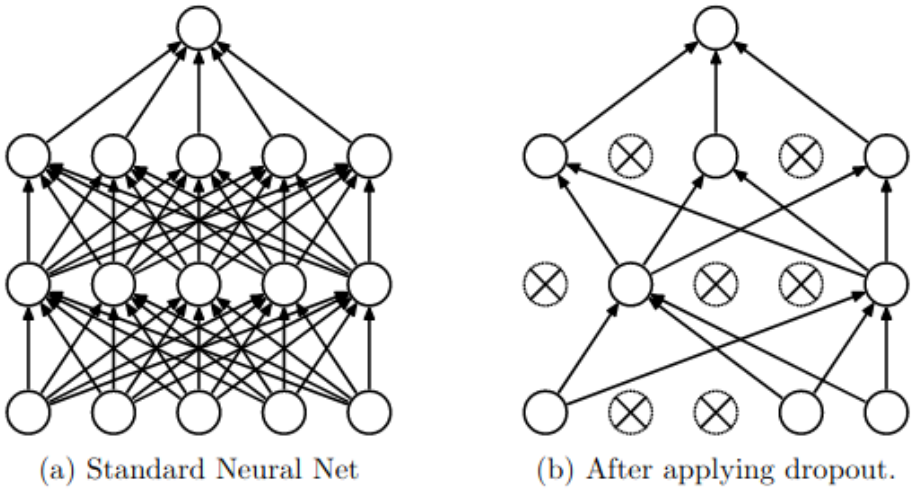

Figura 2.11: Representación visual de *DropOut* (Srivastava et al., 2014b).

2.6.3.3. Normalización por lotes

La normalización de información es un paso casi obligatorio a seguir cuando se prepara un conjunto de datos en un entrenamiento; sin embargo, solo se hace para las neuronas de entrada, lo cual complica el entrenamiento para las neuronas interiores, ya que se requieren unas tazas de aprendizaje bajas y un mayor cuidado con los parámetros iniciales. El término *Batch Normalization* o normalización por bloques, se refiere a hacer de la normalización una parte de la arquitectura de la red y llevarla a cabo en cada mini lote de entrenamiento.

Esto permite mayores tazas de aprendizaje y darle menos atención a la inicialización de los parámetros y en algunos casos llega a evitar el uso de la técnica de *DropOut*. Si comparamos dos modelos distintos, uno el cual utiliza esta técnica y otro que no la utiliza, el modelo que emplea la normalización por lotes obtiene la misma precisión solo que hasta en 14 pasos menos de entrenamiento y superando al otro modelo de forma notoria (Ioffe & Szegedy, 2015).

2.6.3.4. *Global Average Pooling*

En las redes neuronales convolucionales convencionales se utilizan las capas de convolución en las primeras capas de la red, y en las últimas capas los mapas de características son vectorizados y pasan a unas capas totalmente conectadas con activación *softmax* para la clasificación. Sin embargo, las capas totalmente conectadas están propensas al sobre ajuste, lo que provoca una complicación en el entrenamiento del modelo. Con las capas *Global Average Polling* se busca reemplazar las tradicionales capas totalmente conectadas (o *Fully Conected Layers*). A diferencia de otras capas, aquí se toma el promedio de cada mapa de características y se almacena en un vector, el cual se introduce directamente en la capa *Softmax*.

Este tipo de capas tiene algunas ventajas sobre las típicas totalmente conectadas, una de ellas es más nativa de la estructura de convolución, además de que no existe algún parámetro que se deba optimizar, por lo que evitamos aún más el sobre ajuste (Min et al., 2013).

La Figura 2.12 muestra una representación del funcionamiento de las capas *Global Average Polling*.

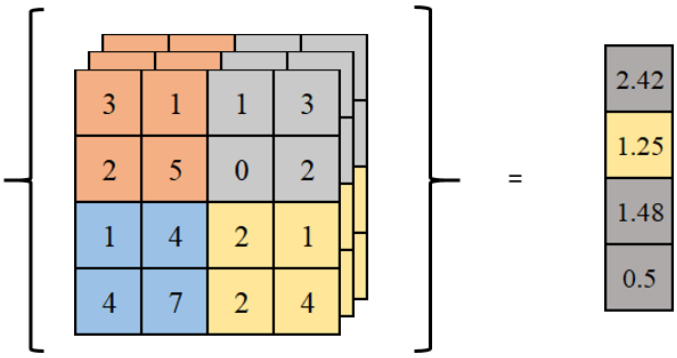

Figura 2.12: Representación *visual Global Average* Pooling

2.6.4. YOLO

You Only Look Once (YOLO, por sus siglas en inglés) (Redmon & Farhadi, 2018) es un sistema de código abierto que se utiliza para la detección de objetos en tiempo real, la principal característica con la que cuenta es que utiliza solamente una red neuronal compleja para realizar la detección en imágenes. Esta tarea se lleva a cabo con redes convolucionales, la configuración de la red está representada en la Figura 2.14, donde se muestra que está constituida por una red de 53 capas de convolución, la cual llamaron Darknet-53. Gracias al proceso de aprendizaje que este algoritmo lleva a cabo, es posible extraer las coordenadas de los objetos detectados en la imagen y su predicción de la clase a la que pertenece. La Figura 2.13 muestra una representación de los datos que se extraen de las imágenes. En la parte izquierda aparece una imagen donde se muestra un vehículo y un cuadro delimitador que hace la función de delimitar la zona donde aparece el vehículo. En la parte izquierda aparece cinco números que representan: en primera posición la clase a la que pertenece el vehículo detectado y los siguientes cuatro números las coordenadas del cuadro delimitador.

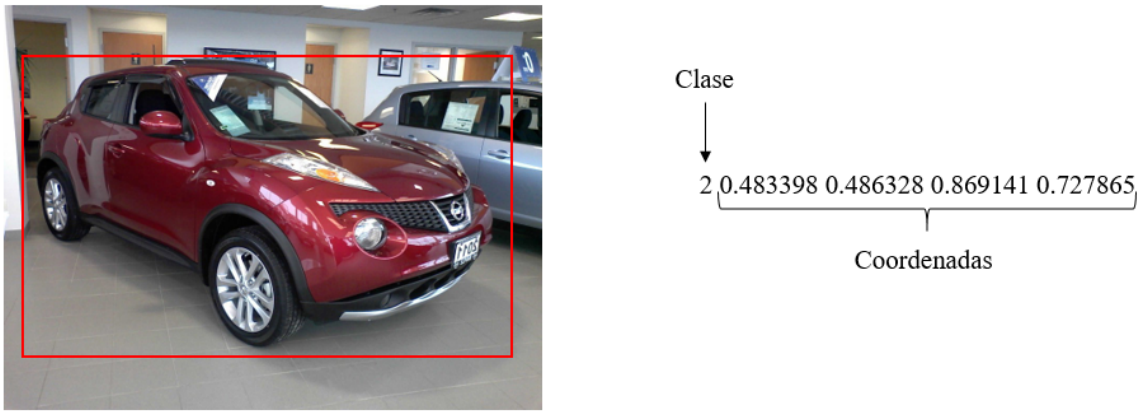

Figura 2.13: Representación de datos extraídos con red YOLO

Layer	Filters size	Repeat	Output size
Image			416 × 416
Conv	32 3 × 3/1	1	416 × 416
Conv	64 3 × 3/2	1	208 × 208
Conv Conv Residual	32 1 × 1/1 64 3 × 3/1	Conv Conv Residual × 1	208 × 208 208 × 208 208 × 208
Conv	128 3 × 3/2	1	104 × 104
Conv Conv Residual	64 1 × 1/1 128 3 × 3/1	Conv Conv Residual × 2	104 × 104 104 × 104 104 × 104
Conv	256 3 × 3/2	1	52 × 52
Conv Conv Residual	128 1 × 1/1 256 3 × 3/1	Conv Conv Residual × 8	52 × 52 52 × 52 52 × 52
Conv	512 3 × 3/2	1	26 × 26
Conv Conv Residual	256 1 × 1/1 512 3 × 3/1	Conv Conv Residual × 8	26 × 26 26 × 26 26 × 26
Conv	1024 3 × 3/2	1	13 × 13
Conv Conv Residual	512 1 × 1/1 1024 3 × 3/1	Conv Conv Residual × 4	13 × 13 13 × 13 13 × 13

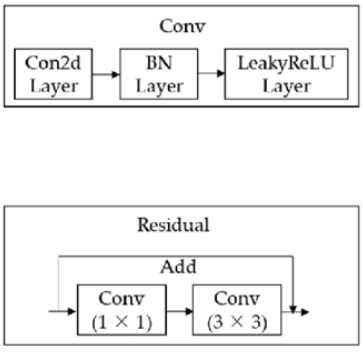

Figura 2.14: Representación de arquitectura de red neuronal de YOLOv3 (Redmon & Farhadi, 2018).

2.7. Filtro de Kalman

El filtro de Kalman (Welch, Bishop et al., 1995) predice el estado de un sistema utilizando control por retroalimentación: el primer paso es estimar el estado futuro y el segundo hacer correcciones de acuerdo a la retroalimentación en forma de ruido o error. Por lo tanto,

las ecuaciones de filtro de Kalman se dividen en dos grupos: ecuaciones de actualización de tiempo y ecuaciones de actualización de medida. Las ecuaciones de actualización de tiempo son responsables de la predicción (en el tiempo) del estado actual y estiman la covarianza para obtener estimaciones pasadas para el próximo paso de tiempo. Las ecuaciones de actualización de medición admiten retroalimentación, es decir, incorporan la nueva medición en la estimación anterior para obtener una estimación posterior mejorada.

La ecuación de actualización de tiempo también se puede considerar como la ecuación de predicción, mientras que la ecuación de actualización de medición se puede considerar como una ecuación de corrección. De hecho, el estimador final es similar al algoritmo de corrección de predicción para resolver problemas numéricos, como se muestra en la Figura 2.15.

Figura 2.15: Representación del ciclo que utiliza el Filtro de Kalman

El filtro de Kalman tiene como objetivo resolver el problema general de estimar el estado $x\epsilon\mathbb{R}^n$ de un proceso controlado en tiempo discreto, el cual se representa con la Ecuación de estado 2.7, y la Ecuación de relación, estado y medición 2.8.

$$x_{k+1} = A_k x_k + B u_k + w_k \tag{2.7}$$

Con una medida $z\epsilon\mathbb{R}^m$:

$$z_x = H_k x_k + v_k \tag{2.8}$$

Las variables w_k y v_k representan el error del proceso y de la medida respectivamente.

El algoritmo de filtro Kalman cuenta con dos fases, la fase predicción (a priori) y la fase de corrección (a posteriori), según se aprecia en la Figura 2.16.

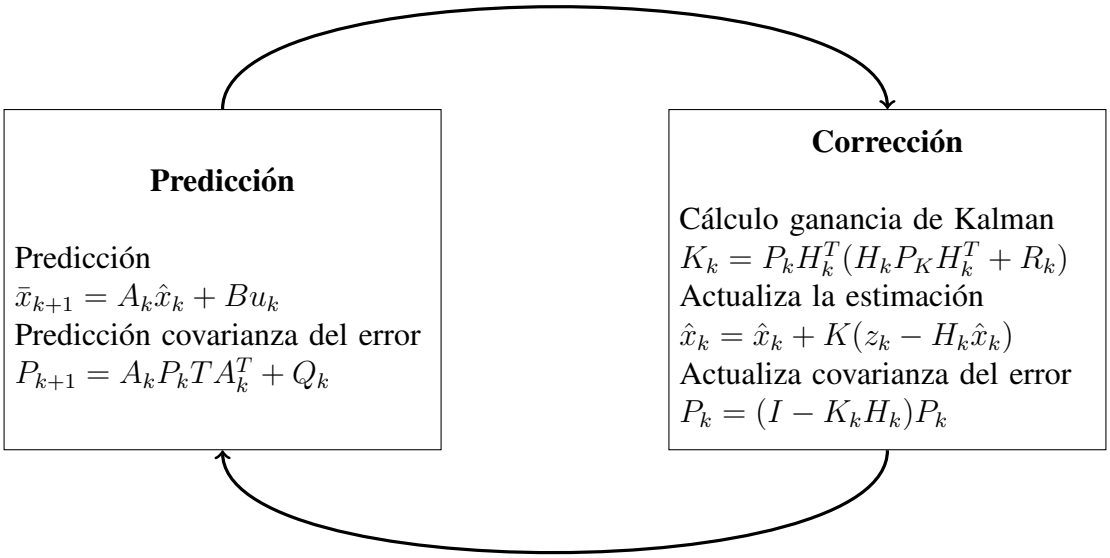

Figura 2.16: Algoritmo de filtro Kalman (Welch, Bishop et al., 1995).

Para la fase de predicción se toman las Ecuaciones 2.9 y 2.10:

$$\bar{x}_{k+1} = A_k\hat{x}_k + Bu_k \tag{2.9}$$

$$P_{k+1} = A_kP_kTA_k^T + Q_k \tag{2.10}$$

Las ecuaciones anteriores pronostican el estado y la covarianza desde k hasta $k+1$. La matriz A representa el estado actual. Q representa la covarianza de la perturbación aleatoria del proceso.

La fase de corrección cuenta con las Ecuaciones 2.11, 2.12 y 2.13:

$$K_k = P_kH_k^T(H_kP_KH_k^T + R_k) \tag{2.11}$$

$$\hat{x}_k = \hat{x}_k + K(z_k - H_k\hat{x}_k) \tag{2.12}$$

$$P_k = (I - K_kH_k)P_k \tag{2.13}$$

La Ecuación 2.11 representa el primer pasa, el cual es el cálculo de ganancia de Kalman, K_t. El siguiente paso es medir el proceso para obtener z, y luego generar una estimación del estado a posteriori incorporando la medición como en la Ecuación 2.12. El último paso es obtener una estimación de la covarianza del error a posteriori mediante la Ecuación 2.13.

Después de cada actualización y medición, el proceso se repite con las estimaciones a posteriori anteriores utilizadas para proyectar o predecir las nuevas estimaciones a priori.

2.8. Tecnologías utilizadas

Para el desarrollo de este proyecto se utilizaron múltiples herramientas, las cuales fueron de apoyo para el manejo de datos, el entrenamiento de los modelos, la elaboración de código, el manejo de versiones, entre estas herramientas está: github, Anaconda, Sublime Text 3, las cuales se utilizaron en el sistema operativo Ubuntu 20.04 (Linux). Además, a continuación se presentan aquellas tecnologías que tuvieron un mayor impacto en el desarrollo de este proyecto.

2.8.1. Python

Python (Rossum, 1994) es un lenguaje de programación de alto nivel interpretado, con el cual se pueden crear aplicaciones de todo tipo. Gracias a ser de código abierto tiene gran popularidad en la comunidad, además de ser un lenguaje fácil de aprender y con gran cantidad de librerías disponibles. Esto ha permitido que sea de gran ayuda en el campo del aprendizaje profundo gracias al desarrollo de librerías como TensorFlow, PyTorch, NumPy, OpenCV, entre otras.

2.8.2. Docker

Docker es una plataforma que nos permite empaquetar aplicaciones en unidades llamadas contenedores, esto con el fin de poder implementar y ajustar dicha aplicación rápidamente en cualquier entorno con la certeza que esta funcionara con éxito. Esto gracias a que en el contenedor se encuentra almacenada todo el código, bibliotecas y herramientas que la

aplicación necesita. Podríamos definir este software como una máquina virtual muy ligera, uno de los objetivos es proporcionar a los programadores un flujo de trabajo, para poder compartir aplicaciones con otros programadores de manera sencilla (Anderson, 2015).

Los contenedores de Docker se pueden utilizar como elemento básico a la hora de crear aplicaciones. Lo anterior debido a que facilita la creación e implementación de arquitecturas de microservicios distribuidos. La implementación de código con canalizaciones de entrega e integración continua estándar. La creación de sistemas de procesamiento de datos altamente escalables, y la creación de plataformas totalmente administradas para sus desarrolladores.

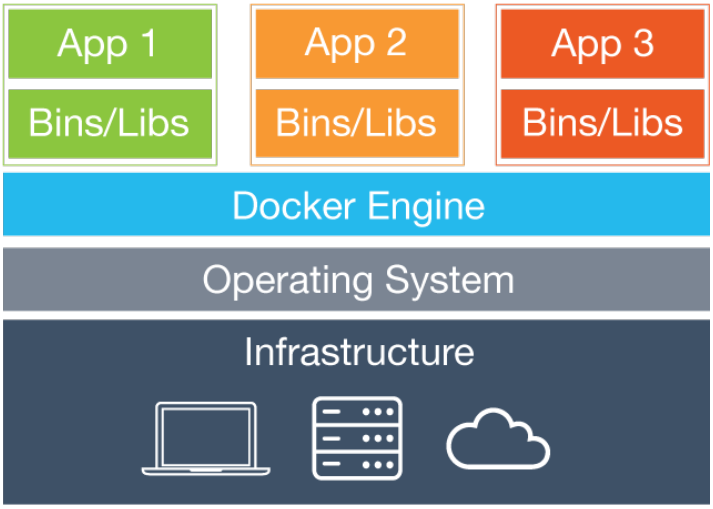

Figura 2.17: Representación de arquitectura de funcionamiento de Docker

2.8.3. Nvidia Docker

NVIDIA Container Toolkit permite crear y ejecutar contenedores acelerados por GPU (*Graphics Processing Unit*). Incluye una biblioteca de tiempo de ejecución de contenedores y utilidades para configurar automáticamente los componentes necesarios para aprovechar las GPU de NVIDIA (Overview s.f.).

Nvidia Docker permite la comunicación entre los contenedores de Docker y las GPU de Nvidia en la máquina utilizada como muestra la Figura 2.18.

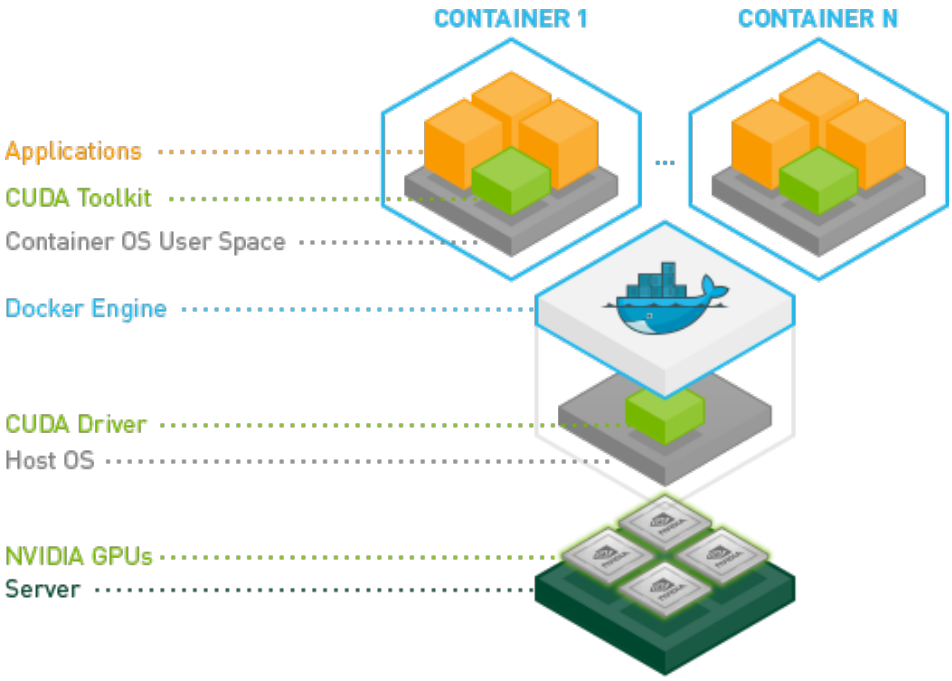

Figura 2.18: Arquitectura de Nvidia Docker (Overview s.f.).

2.8.4. Tensorflow

Una de las librerías de software de código abierto más populares y utilizadas para el aprendizaje profundo es Tensorflow. Una característica es que representa a la computación utilizando gráficos de flujo de datos, estados compartidos y aquellos cálculos que hacen que cambie ese estado. Asigna nodos de gráficos de flujo de datos en varias máquinas, en un clúster y en varios dispositivos informáticos en una sola máquina, como CPU de varios núcleos, GPU de propósito general y ASIC personalizados, conocidos como Unidades de Procesamiento de Tensor (TPU por sus siglas en inglés). Esta arquitectura brinda flexibilidad a los desarrolladores de aplicaciones: mientras que en los diseños anteriores de "servidor de parámetros" se integraba la administración de estado común en el sistema, TensorFlow permite a los desarrolladores probar y desarrollar nuevas optimizaciones y algoritmos de capacitación. TensorFlow admite una variedad de aplicaciones, con un enfoque en el entrenamiento y la inferencia en redes neuronales profundas. Varios servicios de Google usan TensorFlow en producción, y lo hemos lanzado como un proyecto de código abierto y se ha utilizado ampliamente en la investigación de aprendizaje automático (Abadi et al., 2016)).

2.8.5. Conjunto de datos COCO

El conjunto de datos *Microsoft Common Objects in Context* (COCO por sus siglas en inglés) es bastante utilizado en el campo del aprendizaje profundo, debido a que ofrece una gran cantidad de reconocimiento de objetos a lo largo de sus 328,000 imágenes. Existen un total de 80 categorías, y es posible hacer detecciones tanto con segmentación de objetos o con cajas delimitadores, dependiendo de las necesidades de cada aplicación. Algunas de las clases que ofrece este conjunto de datos son: automóvil, motocicleta, autobús, bicicleta y camión. COCO junto con YOLO ofrecen un modelo ya entrenado, el cual es capaz de identificar todas y cada una de las clases implicadas con gran eficiencia. La Figura 2.19 muestra una representación, con imágenes, todas y cada una de las clases que se pueden utilizar con el conjunto de datos MS COCO.

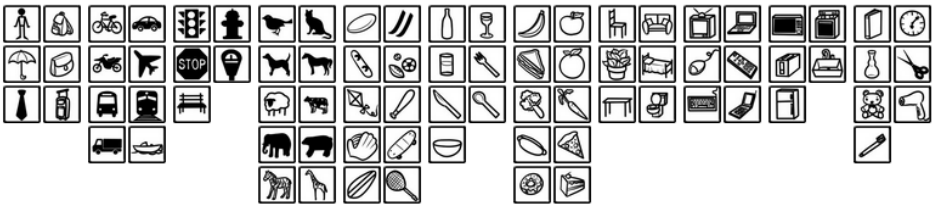

Figura 2.19: Clases incluidas en el conjunto de datos COCO (Lin et al., 2014)

2.8.6. Autotrack

Esta aplicación de código abierto que permite realizar el seguimiento de múltiples vehículos a partir de una serie de imágenes. En su código fuente, el cual es de uso libre, implementa las ecuaciones de Filtro de Kalman, con las cuales hace posible la predicción de la trayectoria de los vehículos. Además de esta función de seguimiento, implementa un algoritmo Húngaro o también llamado Munkres con el cual permite hacer el manejo de las obstrucciones de los vehículos. Esto último lo realiza guardando todas las posiciones de los vehículos detectados en los fotogramas y haciendo comparaciones de todas estas detecciones y determinando las coincidencias que tiene una nueva detección con las ya guardadas (Burnett et al., 2019).

Capítulo 3

Estado del Arte

El presente capítulo muestra una recopilación de trabajos relacionados con sistemas de vigilancia de tráfico vial. Se presentan trabajos que utilizan distintas tecnologías para la recopilación de información vial. Los trabajos se dividen en tres categorías como se mostró en el capítulo 1. Estas categorías son:

- Basados en la carretera: dispositivos se encuentran instalados dentro de la propia calle.
- Basados sobre la carretera: dispositivos se encuentran en instalados en infraestructuras arriba de la carretera o sobrevolándola, a una altura considerable para permitir el paso de todo tipo de vehículo.
- Basados a un lado de la carretera: dispositivos son instalados o colocados, por un lado, de donde transitan los vehículos.

3.1. Sistemas basados en la carretera

En Balid et al., 2018 se lleva a cabo una implementación de un sistema de tráfico inteligente en tiempo real, capaz realizar conteo de vehículos, estimar su velocidad y clasificar de acuerdo a su tamaño, utilizando un par de sensores magnetométricos inalámbricos para enviar la información a un sistema de cómputo. Este estudio obtuvo resultados de 99.98 % en precisión de detección, 97,11 % en precisión de estimación de velocidad y 97 % en precisión de la clasificación.

En Quan et al., 2020 se implementa un sistema parecido al anterior descrito, con la diferencia que se utilizan dos sensores geomagnéticos separados por una distancia de tan solo 8.4 cm. Esto que permite reducir el tiempo de cálculo de la velocidad comparada a tener los sensores con una mayor distancia entre sí como en otros trabajos. La precisión obtenida del cálculo de la velocidad es del 93 %. Este trabajo lleva a cabo solo la recopilación de la velocidad de los vehículos, utilizando los sensores antes mencionados, un transmisor inalámbrico y un modelo de alimentación.

En George et al., 2013 desarrollan e implementan un algoritmo capaz de realizar detecciones, clasificaciones y conteo de vehículos, haciendo uso de un sensor acústico, con el cual se recolecta información en forma de señal acústica. A partir de la recolección de datos se lleva a cabo un procesamiento del audio que comienza con la eliminación de ruidos externos para concentrarse en el provocado por el vehículo. Una vez los audios están preparados, se utiliza dos tipos de redes neuronales, una Red Neuronal Artificial (*Artificial Neuronal Network*, ANN por sus siglas en inglés) y el método K Vecinos Cercanos(*K-nearest Neighbors*, KNN por sus siglas en inglés), para llevar a cabo el proceso de clasificación de la señal. Los vehículos se clasifican en tres clases: pesado, medio y ligero, dependiendo del tamaño del auto que sea la detección entrara en alguna de estas categorías. El resultado en la precisión obtenida fue de 50.62 % para la red de KNN y un 73.42 % en la ANN.

Este trabajo H. Zhao et al., 2018 presenta un sistema de clasificación de vehículos basado en las vibraciones que utiliza la tecnología de Detección Óptica Distribuida de Vibraciones (DOVS por sus siglas en inglés). Describe un método de clasificación completo que incluye el procesamiento de la señal y la extracción de características. Con bajos costes de mantenimiento, este sistema puede recoger datos de clasificación de vehículos a gran escala. En primer lugar, utiliza una fibra de detección incrustada como sensor distribuido para recoger las señales de vibración inducidas por el tráfico. A continuación, extrae varias características de las señales brutas para estimar las configuraciones de los ejes e identificar las categorías de vehículos. Al mismo tiempo, se aplica un método basado en la Descomposición Empírica de Modos (EMD por sus siglas en inglés) para reconstruir las señales con el fin de extraer las características. A continuación, se proponen varios algoritmos de extracción para obtener la configuración de los ejes, la velocidad de desplazamiento y la característica en el dominio

de la frecuencia de cada vehículo. Una vez extraídas todas las características, se diseña un clasificador de varios pasos para clasificar los vehículos en diferentes clases. Además, para evaluar el rendimiento de la clasificación de este sistema, se instaló un sistema prototipo en una carretera de socorro de Shanghai, China, utilizando tecnología de pavimento de hormigón prefabricado. Con una precisión global del 89 %, los resultados de las pruebas muestran un buen rendimiento de este sistema de clasificación.

En este trabajo Huang et al., 2018, se desarrolla un sistema de clasificación de vehículos basado en sensores de rejilla de fibra de vidrio reforzada con polímero en el pavimento (3-D GFRP-FBG). Cuando los vehículos pasan por el pavimento, producen tensiones, que pueden ser monitorizadas por los cambios de longitud de onda del centro de los sensores 3-D GFRP-FBG incrustados. La velocidad del vehículo y la distancia entre ejes se pueden estimar entonces de acuerdo con los diferentes tiempos de llegada de un vehículo a los sitios de los sensores y las velocidades monitoreadas a partir de los cambios de longitud de onda de los sensores en el pavimento. El sistema de clasificación de vehículos de este trabajo utiliza algoritmos de aprendizaje automático de vectores de apoyo para clasificar los vehículos en categorías que van desde vehículos pequeños hasta camiones combinados. Los resultados de las pruebas de campo con tráfico real muestran que el sistema desarrollado puede estimar con precisión las clasificaciones de vehículos con un 98,5 % de exactitud.

3.2. Sistemas basados sobre la carretera

En Ukani et al., 2016 utilizan la sustracción de fondo junto con una red neuronal y máquina de vectores para las tareas de detección y clasificación respectivamente. Un inconveniente de este tipo de técnicas presenta es que el algoritmo no es robusto. Esto que provoca que no sea capaz de trabajar en múltiples escenarios, si existe cambios en la iluminación deberá ser calibrado nuevamente.

En Tang et al., 2017 utilizan una cámara para hacer detecciones y clasificación de vehículos. Para ello, toma las imágenes capturadas y utiliza técnicas de tipo Haar y algoritmos AdaBoost para obtener características de los vehículos y poder localizar la posición en la que se encuentran. A continuación, utiliza la transformada de Wavelet de Gabor y un operador de

patrones binarios locales para extraer aún más características, pero ahora en múltiples escalas y orientaciones. Finalmente, la imagen se divide en pequeñas áreas, de las cuales se extraen y enfocan una serie de histogramas para representar las características del vehículo. Se aplica el análisis de componentes principales para obtener la función de gráfico de baja dimensión, que se utiliza para medir la similitud entre diferentes compuestos en el espacio de Euler, y se extrae la vecindad más cercana para la clasificación final. Los resultados obtenidos muestran una tasa de detección superior al 97 % con una taza de falsos positivos del 3 %, obteniendo también una precisión en el reconocimiento de vehículos arriba del 91 %.

En Mandal et al., 2020 se desarrolla un sistema de videovigilancia a partir de aprendizaje profundo y visión artificial, el cual es capaz de trabajar bajo varias condiciones climatológicas variadas sin verse afectado en gran medida. En la parte del aprendizaje profundo se utilizan redes convolucionales con el objetivo de detectar colas de tráfico, rastrear vehículos estacionados, y llevar un conteo. Aplican la segmentación en pixel para determinar la gravedad de las colas de tráfico. Para la tarea de detección y clasificación, hace una comparación entre dos redes convolucionales populares, *Faster R-CNN* y YOLO, de las cuales la red de YOLO demuestra tener un desempeño superior, aunque no por mucho. Se detectan cinco clases: *Ped, Cyclist, Car, Bus* y *Truck*, obteniendo un promedio de detección del 92 %, mientras que con *Faster R-CNN* un 84 %. En el apartado del seguimiento se utilizan dos algoritmos IOU (Bochinski et al., 2017) y *Feature Tracker*, los cuales hacen comparaciones entre sus cálculos y permiten el manejo de obstrucciones.

En Anil Rao et al., 2015 se implementan un sistema de monitoreo de tráfico utilizando una cámara junto a un algoritmo de sustracción de fondo para la tarea de detección y clasificación. Además de utilizar el filtro de Kalman y el algoritmo húngaro para la tarea de seguimiento, así mismo utiliza una transformación de la perspectiva de las imágenes para obtener distancias recorridas de los vehículos en píxeles. Posteriormente, realizar la conversión a datos reales con los cuales poder calcular la velocidad. El objetivo principal fue obtener la velocidad de los vehículos, ya que no se obtiene ningún tipo de clasificación después de hacer la detección. Los resultados muestra un error de +-3 km/h.

En Luvizon et al., 2014 se presenta un trabajo enfocado a la obtención de la velocidad de vehículos, para ello hacen detecciones y seguimiento con distintos algoritmos. En la tarea de

detección utilizan un reconocimiento de caracteres llamado *Snoopertext*, el cual lleva a cabo un procesamiento de imágenes iniciando con una segmentación de la imagen dejando visible solo la parte de la placa vehicular. Posteriormente, pasan a un filtrado con el cual se detectan los caracteres de forma más definida y por último se realiza una agrupación de todos los caracteres, esta detección se hace con el propósito de obtener ciertas características del vehículo y poder detectar la zona en la que se encuentra y comenzar a realizar su seguimiento. Para ello se utiliza el algoritmo KLT (Suhr, 2009) en conjunto con el algoritmo SIFT (Lowe, 2004). La función de estimación de la velocidad lleva a cabo una transformación de perspectiva de los fotogramas junto con cálculos para obtener la distancia, los cuales son apoyados por los algoritmos de seguimiento. Como resultados obtenidos obtuvieron un error de - 3.24km/h y + 3.91km/h, cabe señalar que solo se lleva a cabo detección, seguimiento y estimación de la velocidad.

En Zhang et al., 2017 se utiliza una red neuronal de convolución para llevar a cabo la tarea de detección y clasificación. Esta implementación se lleva a cabo con cámaras de video frontal que se encuentran instaladas en vehículos en movimiento. La tarea de clasificación se lleva a cabo con el conjunto de datos KITTI (Geiger et al., 2013). A diferencia de otros trabajos, aquí se utilizan dos redes neuronales, la primera, una R-CNN detecta objetos relevantes como coches y peatones, de los cuales extrae información de la posición en la que se encuentran y esta información se utiliza de entrada en una CNN. Con esta red se obtienen tres propiedades importantes de movimiento propio de cada vehículo, que son: velocidad de avance, aceleración hacia adelante y velocidad angular. Cabe señalar que no se lleva a cabo un seguimiento del vehículo como tal, sino que en cada fotograma se va calculando las tres propiedades a los vehículos tomando en cuenta el flujo de movimiento. Como resultados se muestra un MSE de 6.8 para la velocidad, un 0.366 para aceleración y un 0.012 para la velocidad angular.

En Kampelmühler et al., 2018 se desarrolló un trabajo el cual fue la propuesta ganadora en el reto de estimación de velocidad de vehículos CVPR2017. Este trabajo implementa una cámara instalada en un coche en movimiento y se busca estimar la velocidad relativa de los vehículos frente a él con el objetivo de evitar colisiones. La tarea de seguimiento de vehículos la lleva a cabo con dos herramientas que proporciona OpenCV $MedianFlow$ y MIL,

ambas implementaciones, ofrecen la posibilidad de adaptarlas cajas de seguimiento durante la trayectoria. Sin embargo, esos algoritmos sin inestables cuando se presentan obstrucciones. Para la tarea de estimación de la velocidad, implementa una red de perceptrón multicapa (MLP por sus siglas en inglés) de regresión. Al final de la implementación se obtuvo un error promedio de 4.032 km/h.

En Niu et al., 2018 utiliza un Vehículo Aéreo no Tripulado (UAV por sus siglas en inglés) para llevar a cabo la tarea de vigilancia de tráfico. Para llevar a cabo la tarea de detección se implementa un algoritmo *Haar cascade* el cual se entrenó con 3750 imágenes obtenidas del UAV en diferentes áreas. La función de seguimiento la realizaron tomando en cuenta el límite de velocidad de la zona monitoreada. De esta forma se conoce la distancia máxima la cual un vehículo puede recorrer de un fotograma a otro. Por lo que al detectar un vehículo en un fotograma se compara su posición con todos los demás vehículos anteriormente detectados y si se obtiene el vehículo con el que menos distancia tiene de separación. De esta forma se dice que es el mismo vehículo y se hacen las actualizaciones de datos correspondientes. Los resultados muestran que se obtuvo una precisión de detección entre 84 % y 90 %.

3.3. Sistemas basados a un lado de la carretera

En J. Zhao et al., 2019 desarrollan un sistema que detecta vehículos y peatones utilizando sensores LiDAR, para ellos se utilizaron un total de 16 dispositivos láser y de comunicación. Estos se utilizaron para llevar a cabo la recolección de datos del entorno. Para la agrupación de datos se utiliza un método denominado DBSCAN modificado, junto con un diseño de división del rango de detección en subáreas basadas en la distancia del sensor para mejorar la precisión y reducir el coste computacional. Para llevar a cabo la clasificación de objetos, se utiliza una red neuronal de retropropagación, observando que la dirección de la distribución de los puntos agrupados era la clave para distinguir los grupos de peatones y vehículos cuando los datos LiDAR que son recogidos no pueden llevar a cabo una descripción fina de los objetos. En la tarea de seguimiento implementaron Filtro de Kalman discreto y un modelo de asociación de objetos basado en la distancia. Una vez integrados los componentes, los resultados de detección y seguimiento alcanzaron un resultado promedio de 95 % de precisión.

3.4. Comparación de trabajos desarrollados

La Tabla 3.1 muestra comparaciones de distintos factores que son utilizados en los trabajos del estado del arte. Los aspectos que se comparan son: conjunto de datos, datos de entrada, características extraídas, consideraciones ambientales y limitaciones. Cada autor utiliza técnicas y métodos distintos y toma en cuenta diferentes circunstancias que hacen a su trabajo único. Dependiendo de la técnica que se utilizan usaran o no un conjunto de datos, también dependiendo de la tecnología empleada los datos de entrada cambian. Se muestran también de las características que se extraen en cada trabajo. También se muestran dos aspectos importantes que son las consideraciones ambientales y limitaciones. Estas se deben tomar en cuenta desde un inicio, ya que son necesarias para decidir la posible solución en esas condiciones. Esta tabla también resume que para resolver la problemática de extracción de características se deben abordar distintas áreas y tecnologías.

Tabla 3.1: Tabla comparativa entre trabajos del estado del arte

Autor	Año	Conjunto de datos	Datos de entrada			Caracteristicas extraidas	Consideraciones ambientales	Limitaciones
			Video	Señal sensores	Tensiones			
Balid	2018	No aplica		X		Tiempo entre señales	Sensores incrustados en carretera	Clasificación solo por tamaño
Quan	2020	No aplica		X		Tiempo entre señales	Sensores incrustados en carretera	Solo realiza calculo de velocidad
George	2013	No aplica		X		Ruido provocado por vehiculo	Se eliminan ruido ajeno a vehiculo	El ambiente puede contener ruido excesivo
H. Zhao	2018	No aplica		X		Perturbaciones en señales	Utiliza ormigon especial para instalar fibra de deteccion	Tiempo de instalacion elevado
Huang	2018	No aplica			X	Cambios de longitud de onda	Los sensores van incrustados en la carretera	Clasificación solo por tamaño
Ukani	2016	No aplica	X			coordenadas de objetos en movimiento	Se debe calibrar cuando se presentan cambios en la iluminacion	No es un sistema robusto
Tang	2017	No aplica	X			Coordenadas de vehiculos	Dificultades en escenas nocturnas	Los algoritmos utilizados son afectados por la iluminacio
Mandal	2020	Personalizado 18509 imagenes	X			Coordenadas de vehiculos deteccion de colas de trafico	Aplicable a grandes areas	Se necesita camara de alta resolucion para evitar imagenes pixeleadas
Anil Rao	2015	No aplica	X			Pixeles recorridos por el vehiculo	Utiliza la homografía para la transformación de la perspectiva	Calibración de cámara de ser calibrada manualmente y ser monitoreada
Luvizon	2014	No aplica	X			Caracteres de placa vehicular	Rectificación de imagenes	Problemas de detección si la placa del auto no es muy legible o es muy pequeña
Zhang	2017	KITTI	X			Coordenadas del vehiculo	No tiene consideraciones	Los umbrales de detección deben ser calibrados manualmente
Kampelmuhler	2018	1074 sequences in freeway traffic	X			Rutas de vehículos, profundidad y movimiento	No tiene consideraciones	Tiempo de procesamiento para la extracción de características.
Niu	2018	Personalizado 3750 de UAV	X			Segmentación de objetos en movimiento	Se toma en cuenta el limite de velocidad de la zona	El UAV no puede estar todo el día monitoreando
Este trabajo	2022	COCO y conjunto de datos propio	X			Distancia y tiempo	No tiene consideraciones	Los vídeos deben ser analizados manualmente para separar las muestras malas

Capítulo 4

Metodología

En el presente capítulo se presentan las etapas de desarrollo que se utilizaron para darle solución al proyecto. En la Figura 4.1 se muestran las etapas realizadas durante la elaboración de este trabajo. Se aborda desde el análisis de requerimientos, seguido de la etapa de entrenamientos y por último el desarrollo del sistema extractor de características.

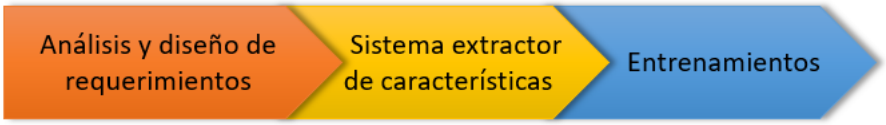

Figura 4.1: Etapas de desarrollo

- Análisis y diseño del sistema.- en esta etapa se propone el diseño para que las herramientas de extracción, seguimiento y clasificación trabajen en conjunto.
- Sistema extractor de características.- en esta etapa se explican las funciones que en conjunto extraen información vial.
- Entrenamientos.- en esta etapa se muestra el proceso para llevar a cabo los entrenamientos del modelo clasificador.

4.1. Análisis y diseño del sistema

Los requisitos funcionales representan las funcionalidades con las que el proyecto debe contar, estos se construyen a partir de los objetivos y posibles situaciones que se presenten al

estar utilizando la aplicación resultante. A continuación se muestran los requisitos funcionales de este trabajo

- RF-01: el sistema deberá hacer detección de vehículos
- RF-02: el sistema deberá clasificar vehículos en distintas clases
- RF-03: el sistema deberá realizar seguimiento a los vehículos detectados
- RF-04 el sistema deberá hacer conteo de vehículos
- RF-05: el sistema deberá determinar la velocidad de los vehículos
- RF-06: el sistema guardará imágenes de vehículos detectados
- RF-07: el sistema deberá guardar la información obtenida en una base de datos

A continuación se enlistan los requisitos de calidad que se consideran para el desarrollo del sistema:

- RC-01: el sistema deberá ser capaz de trabajar con CPU o GPU.
- RC-02: el sistema deberá ser capaz de procesar archivos tipo .mp4 y .MOV.
- RC-03: el sistema deberá tener una precisión de detección por encima del 90 %.

4.1.1. Actores

Los actores son aquellas personas, sistemas o dispositivos los cuales interactúan con el sistema. A continuación se muestran:

- Usuario: se encarga de utilizar el sistema para indicar que archivos procesar

4.1.2. Casos de uso

Los caso de uso son descripciones de las actividades que deben realizarse por alguien, para llevar a cabo un proceso. Son creador a partir de los requisitos funcionales, y se presentan a continuación:

La Tabla 4.1 se presentan los casos de uso que se pueden presentar al utilizar la aplicación.

Tabla 4.1: Casos de uso

Casos de uso	Descripción
CU-01	Cargar archivo de video para análisis
CU-02	Consulta de expediente

En el CU-01 se indica que el usuario puede llevar a cabo la selección del archivo que se pretende analizar por el sistema. Así mismo, el CU-02 indica que se puede hacer consulta al expediente generado por el sistema.

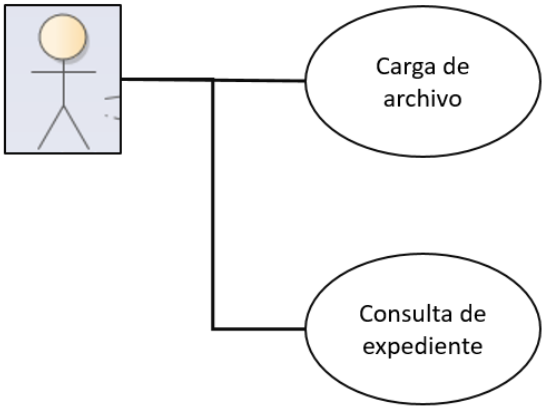

Figura 4.2: Diagrama de casos de uso

4.1.3. Diagrama de contexto

La Figura 4.3 muestra el diagrama de contexto del sistema. En la parte superior se muestra al usuario que utiliza el sistema. En la parte de en medio se muestra al sistema y a los dispositivos de hardware que necesita para poder trabajar. Y por último, en la parte inferior aparece la base de datos donde el sistema almacena la información extraída, además de la visualización de resultados que genera el sistema.

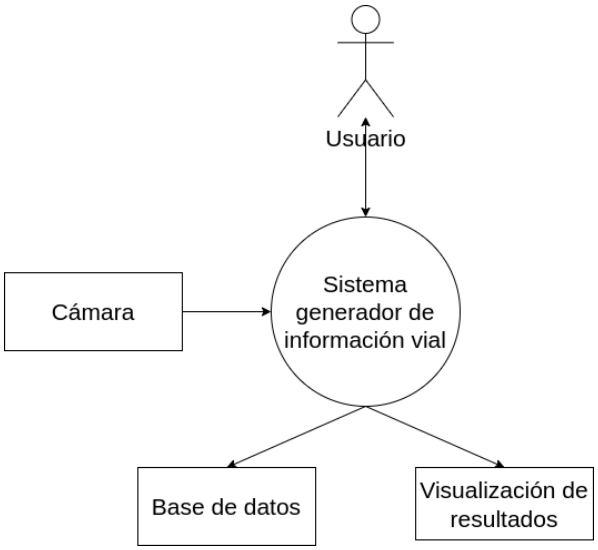

Figura 4.3: Diagrama de contexto

4.1.4. Arquetipos

Los arquetipos se definen con base en las interacciones que los actores tienen con el sistema para cumplir con los requisitos. A continuación son descritos:

- Usuario.- es una representación de la persona que utiliza al sistema.
- Video.- es una representación del archivo .mp4 o .MOV.
- Fotograma.- es una representación de las imágenes que se extraen del archivo indicado por el usuario.
- Vehículo.- es una representación de los datos que son extraídos por cada vehículo en los fotogramas.
- Almacenamiento.- es una representación del archivo que genera el sistema con la información extraída.

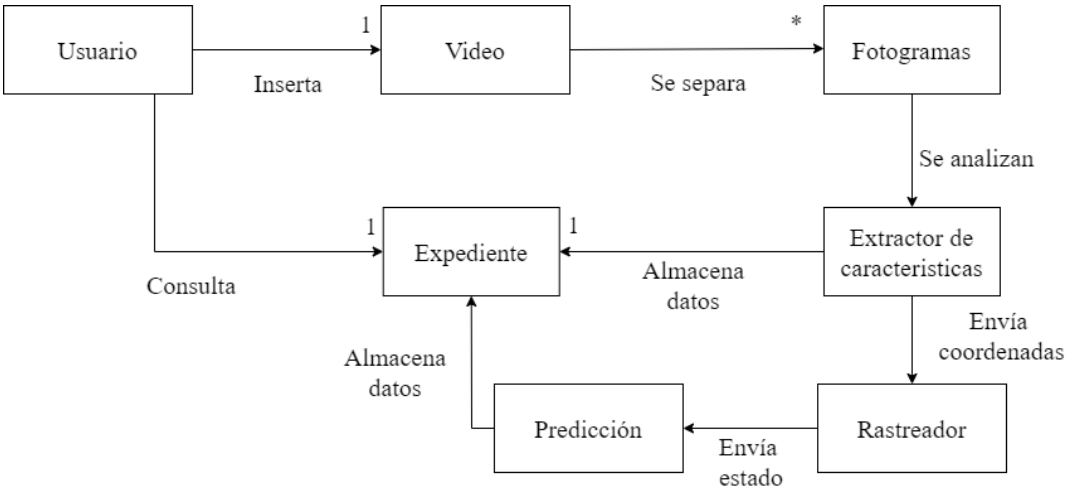

Figura 4.4: Representación de la interacción de los arquetipos

En la Figura 4.4 se puede observar la interacción de los arquetipos del sistema entre sí. La interacción inicia cuando se extraen los fotogramas del archivo de video otorgado. Estos fotogramas son tratados por un modelo el cual hace las detecciones y clasificaciones correspondientes. Con la información generada por el modelo se genera un rastreador a cada vehículo, y por último toda la información generada es almacenada en un expediente, el cual se puede consultar posteriormente.

4.1.5. Arquitectura

El proceso de análisis del extractor de características se muestra en la Figura 4.5. Aquí se observa que se inicia con un archivo de video, el cual puede ser formato .mp4 o .MOV. El primer proceso que interviene es la extracción de fotogramas, los cuales son enviados al detector. El detector genera los cuadros delimitadores que se envían al clasificador para obtener la clase del objeto detectado. Una vez hecha la clasificación se envían coordenadas al rastreador, donde se guarda la posición y calcula variables de estado del vehículo que requiere el Filtro de Kalman para realizar correcciones. Tales correcciones se realizan en el predictor. Además, en el rastreador extraen características como distancia, tiempo y conteo de vehículos, las cuales son guardan en el almacenamiento. Una vez obtenidos los datos en el almacenamiento, se genera un archivo CSV. Este puede ser consultado una vez terminado el proceso de análisis del video. A continuación se describen las estradas y salidas de los

procesos indicados en la arquitectura.

- Extractor de fotogramas.- recibe un archivo de video, el cual separa en fotogramas, y en su salida entrega una matrices de datos correspondientes a los fotogramas del video.
- Detector.- recibe una matriz de datos correspondientes a los fotogramas extraídos, y en su salida entrega las coordenadas pertenecientes a los cuadros delimitadores de la región de interés donde se encuentra a un vehículo.
- Clasificador.- cuenta con una entrada y dos salidas. En su entrada recibe las coordenadas de los cuadros delimitadores. En su primer salida envía una lista con las coordenadas y la clase a la que se asignó el objeto dentro del cuadro delimitador. En su segunda salida envía los datos de clasificación de los objetos detectados al almacenamiento
- Rastreador.- cuenta con dos entradas y dos salidas. En su primer entrada recibe coordenadas de objetos detectados. En su primera salida envía el estado actual del vehículo. En su segunda entrada recibe la predicción de la posición del vehículo. En su segunda salida envía características extraídas.
- Predictor.- recibe en su entrada el estado actual del vehículo y en su salida envía una predicción de la posición futura del vehículo.
- Almacenamiento.- cuenta con dos entradas y una salida. En su entrada recibe una lista de los vehículos, y sus características, que cumplieron con la condición de haber cruzado por el punto A y el punto B. En su salida entrega un archivo CSV donde se guardan las características de los vehículos.

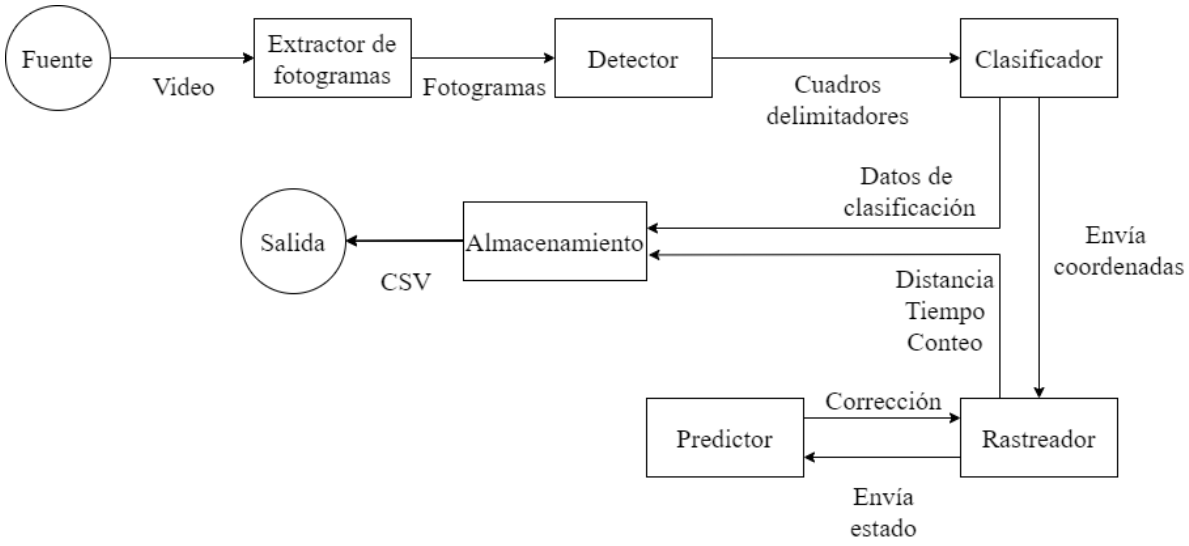

Figura 4.5: Representación del proceso de extracción de información vial

4.2. Sistema extractor de características

A continuación se presenta el funcionamiento de los procesos que se encargan de extraer información en el sistema extractor de características. Cada proceso se encarga de una tarea de transformación de información distinta para enviarla al siguiente e ir almacenándola. Gracias al trabajo en conjunto de los procesos se extraen datos de información vial y se guardan en un archivo.

4.2.1. Extracción de fotogramas

Este es el proceso inicial donde la información se comienza a preparar, para extraer características de ella. Se comienza con un archivo de video en formato .mp4 .MOV, el cual cuenta con unas dimensiones de 1920 x 1080 píxeles y una velocidad de 60 fotogramas por segundo, es decir, cada segundo está compuesto de 60 imágenes con las mismas dimensiones. Estos fotogramas son las imágenes que el sistema necesita para comenzar a hacer las detecciones y clasificaciones de vehículos. En la Figura 4.6 se puede observar una representación visual de la extracción de fotogramas. En ella se muestra que se le proporciona una archivo de video al sistema y este realiza una separación de fotogramas y posteriormente convierte dichos fotogramas a matrices. Cada valor de la matriz representa el valor de un pixel de la imagen, este

puede ir de 0 hasta 255.

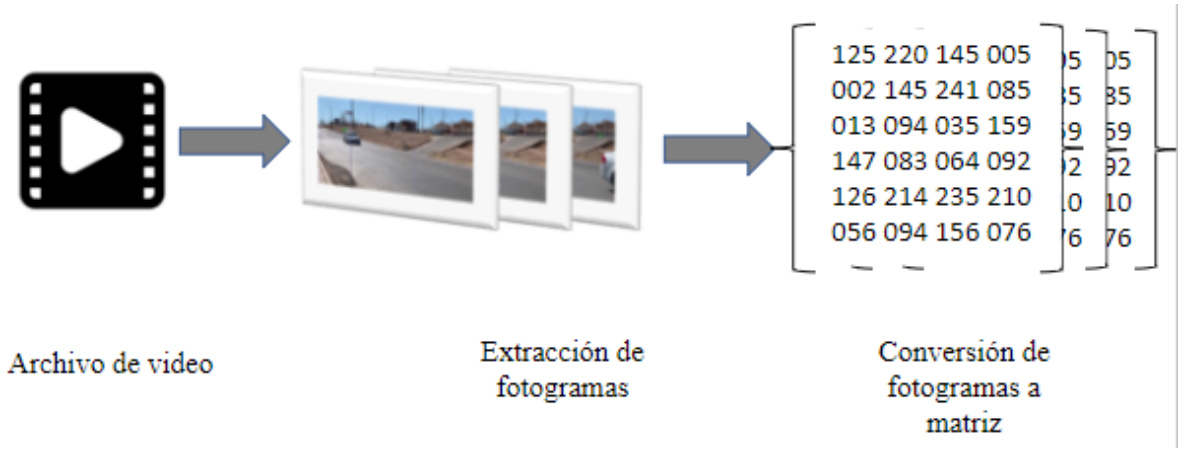

Figura 4.6: Representación de extracción de fotograma

Después del proceso anterior, se lleva a cabo una normalización de datos para obtener números más pequeños y sencillos de procesar. Se toma la matriz de cada fotograma y cada pixel se divide entre 255. De esta manera los números resultantes van de cero a uno. En la Figura 4.7 se muestra el resultado de esta normalización.

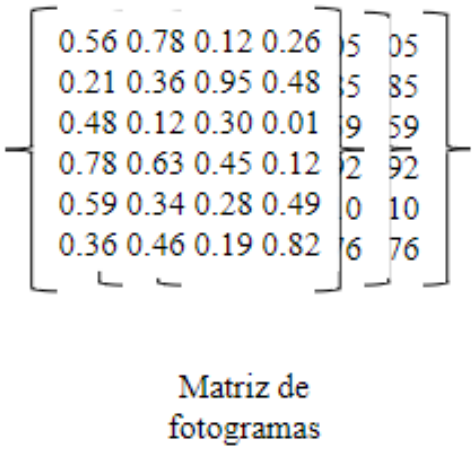

Figura 4.7: Representación de matriz de fotogramas con píxeles normalizados

4.2.2. Detección y clasificación

Para poder llevar a cabo las detecciones y clasificaciones, se utilizan las matrices de datos normalizadas que se extraen en la sección anterior. Cada matriz se procesa a través del modelo

clasificador, el cual tiene la función de detectar y clasificar. y calcular las coordenadas en las que los vehículos se encuentran en cada fotograma.

En la Figura 4.8 se muestra una representación del flujo de los datos al realizar la tarea de clasificación. Durante el proceso de clasificación del modelo, este realiza una serie de transformaciones de datos por sus distintos tipos de capas y filtros. Al finalizar entrega como salida una lista con número que representan datos extraídos tanto de la detección como la clasificación de los vehículos en la imagen, en caso de haber uno o más. Para cada vehículo detectado se obtienen cinco números, y estos representan:

- En primera posición: número de clase a la que se detecta perteneciente el vehículo.
- De segunda a quinta posición: representan a las coordenadas de las esquinas del cuadro delimitador que se utiliza para señalar la posición de un vehículo en el fotograma.

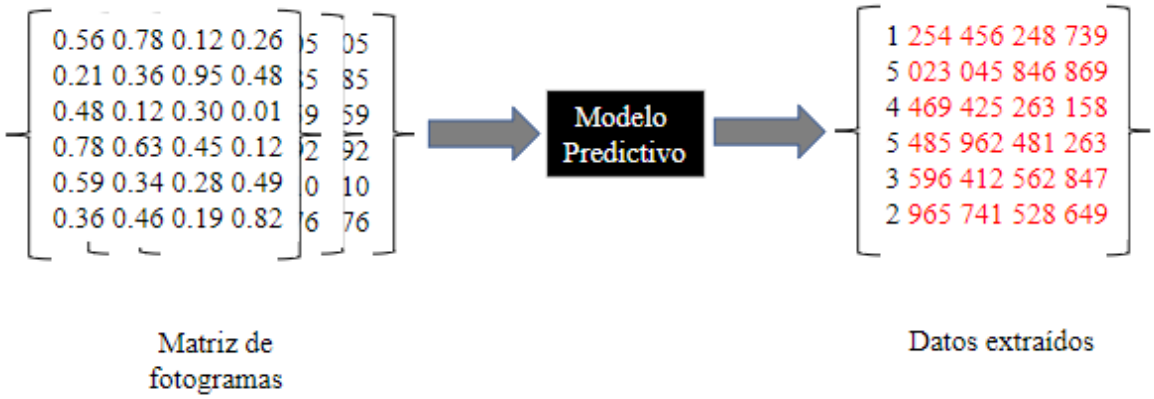

Figura 4.8: Funcionamiento de modelo predictivo

En caso de que en el fotograma haya más de un vehículo, el modelo predictivo guarda la información de todos ellos en una lista. La Figura 4.8 muestra un ejemplo, es una imagen donde se realizaron seis detecciones. En la parte derecha aparecen los datos de dichas detecciones. En primera posición y color negro la clase a la que pertenece y en las siguientes posiciones, y color rojo, las coordenadas de los cuadros delimitadores en el fotograma. a la cual podemos acceder para visualizar su contenido y manejar los datos, además se utiliza más adelante.

4.2.3. Seguimiento, frecuencia y reapariciones

La tarea de seguimiento se lleva a cabo con filtro de Kalman, el cual se utiliza para hacer predicciones de la posición del vehículo en el siguiente fotograma. Además, se guardan todas las posiciones de los objetos detectados para dibujar una trayectoria de recorrido de estos. Se implementa regresión lineal para dibujar la trayectoria de predicción de los vehículos. Estas trayectorias se muestran en la Figura 4.9. En la misma se muestran dos líneas verticales de color azul, estas líneas las utilizamos para indicarle al sistema los puntos de entrada y salida de los vehículos. A estas les llamaremos punto A, izquierda, y B, derecha. Estos dos puntos deben ser cruzados por los vehículos para que sean considerados en el archivo de información.

Figura 4.9: Detección de vehículos en punto B, sus trayectorias y triangulación del generado con el punto A y B.

Este proceso de seguimiento inicia recibiendo las coordenadas del vehículo en el fotograma, luego aplica sus ecuaciones y realiza la predicción de la posición del vehículo en el próximo fotograma. Cuando se analiza el fotograma siguiente y se comienza a calcular la próxima posición, se utilizan los resultados anteriores para hacer una corrección en los cálculos actuales, con el propósito de obtener cada vez mejores resultados.

Cuando se realiza una nueva detección de vehículo en un fotograma, esta se agrega a una lista de detecciones, las cuales se asignan o un rastreador. En caso de que el vehículo haya

sido detectado anteriormente, no se le asigna un nuevo rastreador, sino que se obtiene un parámetro denominado IOU. Este parámetro determina que el vehículo ya ha sido detectado anteriormente y, por lo tanto, ya cuenta con un rastreador. La librería Tensorflow cuenta con una función para calcular el valor de este parámetro y este es un cálculo que se obtiene a partir de pasos de tiempo que ha dado el vehículo.

4.2.4. Conteo vehicular

El conteo vehicular se debe realizar tomando en cuenta reapariciones de vehículos cuando estos hayan sido obstruidos y no captados por la cámara en algún momento. Para llevarlo a cabo se apoya en algunas funciones del proceso de reaparición de vehículos, la cual se explicó anteriormente. El Algoritmo 4.1 muestra el proceso para llevar a cabo el conteo de vehículos. Este algoritmo recibe una lista de detecciones que se realizan en el fotograma analizado. En un inicio el contador inicia en cero y con las detecciones nuevas va aumentando. El proceso trata las detecciones una por una, calcula el parámetro IOU y después hace una comparación con las detecciones hechas anteriormente para determinar si es una nueva detección o una ya hecha anteriormente. En caso de que la detección haya sido nueva, se aumenta el contador en uno, esto se repite hasta terminar con las detecciones en el fotograma y se retorna el contador.

Algorithm 4.1 *VehicleCounting (Data)*

```
Data: Detections_list
 0 ⟶ Counter
 Detection ⟵ First(Data)
 while Detections do
 |  Calculate_IOU (Detection) ⟶ Parameter_IOU ;
 |  Comparison (Parameter_IOU) ⟶ New_Detection
 |   if New_Detection then
 |   |  1+Counter ⟶ Counter
 |   end
 |
end
 Return Counter
```

4.3. Preparación de entrenamientos

Para el proceso de entrenamiento con nuestro conjunto de datos se utiliza *DarkNet* (Wang et al., 2021). Este es un *FrameWork* de redes neuronales de código abierto, el cual permite entrenar tanto con CPU como GPU. Ofrece la posibilidad de crear modelos predictivos personalizados de imágenes. Para ello se deben instalar paqueterías como OpenCV, CUDA, CUDNN, en caso de querer trabajar con GPU, lo cual hace que el tiempo de entrenamiento sea mayor hasta 100 veces menos a comparación de trabajar con CPU. En caso de solo contar con CPU, solo se necesita OpenCV. Esta red neuronal permite hacer *Transfer Learning*, para ello se utilizan pesos pre-entrenados de YOLOv3.

Esta red cuenta con algunos años de desarrollo, durante este tiempo han implementado algunas versiones y mejoras. Debido a ello, los parámetros que se modifican en la red son unos cuantos, y se muestran a continuación:

- *Batch*.- cantidad de imágenes en un lote que son procesadas por la red neuronal.
- Épocas.- cantidad iteraciones que realiza la red neuronal procesando el conjunto de datos.
- Tamaño de imágenes.- el *framework* cuenta con la función de redimensionar imágenes. Se puede escoger el tamaño de las imágenes, la única restricción es que el tamaño debe ser múltiplo de 32.

4.3.1. Preparación de imágenes

Los autores de *DarkNet* recomiendan el uso de al menos 1000 imágenes por clase para obtener buenos resultados de clasificación. Debido a que el entrenamiento es supervisado, las imágenes deben contar con una etiqueta de datos que indique la clase a la que pertenece el objeto en ella. Esta etiqueta es un archivo de texto que debe contar con el formato que se muestra en la Figura 4.10. De lado izquierdo se encuentra la imagen a etiquetar y del lado derecho dos líneas de números. Cada línea corresponde a un objeto dentro de la imagen. El primer numeró representa el identificador de la clase en la que se clasifica el objeto detectado

y los siguientes corresponden a las coordenadas del un cuadro delimitador que indica donde se encuentra el objeto en la imagen.

1 192.64 500.45346500000005 368.64 573.503235
1 491.52 505.58000499999997 748.8 603.6199449

Figura 4.10: Representación de formato de etiqueta de datos

Capítulo 5

Resultados

En el presente capítulo se presentan los resultados obtenidos con base en el desarrollo del sistema extractor de características viales. En él se muestran tiempos de procesamiento y los datos extraídos del video. Además, se presentan los resultados obtenidos con los distintos entrenamientos realizados con las redes convolucionales de YOLOv3. En cada entrenamiento realizado se hicieron modificaciones en los hiperparámetros del entrenamiento, permitiendo mejorar los resultados conseguidos en un inicio.

5.1. Caso de estudio

Para comenzar con el procesamiento se utilizaron las muestras obtenidas de un trabajo anterior al presente. Estas muestras fueron tomadas en la Avenida Las Torres en la ciudad de Culiacán, Sinaloa, México, en dos distintos horarios, el primero de 10:00 a 12:00 horas y el segundo de 17:00 y las 19:00. Ante esto, se presentó un mayor flujo vehicular en el horario de la tarde.

El lugar de toma de muestras presenta un flujo vehicular de oeste a este, por lo que la dirección de los vehículos es de derecha a izquierda para la cámara. También se observa que la calle por la que pasan los vehículos no está nivelada, sino que presenta una inclinación. La cámara se instaló en un punto bajo de esta inclinación, por lo que los vehículos circulan de arriba abajo. Para la instalación de la cámara se consideró que esta no estuviera paralela al flujo vehicular, por lo que se colocó apuntando hacia arriba de la avenida.

En la Figura 5.1 se muestra una imagen del lugar donde se llevó a cabo la toma de mues-

tras. También es la posición en que se instaló la cámara, se aprecia la inclinación de la avenida y la posición en que la cámara apunta hacia el flujo vehicular.

Figura 5.1: Lugar donde se tomaron las muestras

5.2. Entrenamientos de modelo clasificador

A continuación se muestran los resultados obtenidos en el proceso de entrenamiento de modelos predictivo de vehículos. En este se entrenó un modelo con un total de, 21000 imágenes. 14000 imágenes se utilizaron para entrenamiento y las 7000 restantes se utilizaron para las pruebas del modelo.

5.2.1. Conjunto de datos

Se llevó a cabo una recopilación de imágenes para llevar a cabo los entrenamientos del modelo predictivo, en este conjunto de datos se consideraron las clases: auto, camioneta, autobús, van, motocicleta, bicicleta y persona. Se recopilaron alrededor de, 21000 imágenes entre todas las clases, y un aproximado de 3000 imágenes por clase. En la Tabla 5.1 se muestra la cantidad de imágenes por clase que conformaron al conjunto de datos.

Tabla 5.1: Cantidad de imágenes del conjunto de datos por clase

Clase	Cantidad de imágenes
Auto	3000
Camioneta	3000
Autobús	3000
Van	3000
Motocicleta	3000
Bicicleta	3000
Persona	3000

5.2.2. Preparación de conjunto de datos

Una vez obtenidas las imágenes que se utilizan en el entrenamiento del modelo, se debe llevar a cabo un proceso de etiquetado e identificación, esto debido a que el entrenamiento es de tipo supervisado. Por lo que se debe de indicar en cada imagen los objetos que aparecen, así como las coordenadas donde se encuentran posicionados. Este proceso se lleva a cabo con un software llamado LabelImg (Tzutalin, 2015), el cual nos ofrece la posibilidad de etiquetar las imágenes. Esto es, indicando manualmente por medio de un cuadro delimitador la posición en que se encuentra el objeto en la imagen, posteriormente el software muestra la lista de las clases que se configuraron para etiquetar, y se selecciona la clase a la cual pertenece el objeto. Automáticamente, este programa va creando un archivo de texto adicional a cada imagen con su mismo nombre, en el cual aparecen los datos de cada objeto indicado en la imagen, cada línea representa los datos de un objeto, y en cada una aparecen cinco números, los cuales representan:

- En primera posición: número de clase a la que se detecta perteneciente el vehículo.
- De segunda a quinta posición: representan a las coordenadas de las esquinas del cuadro delimitador que se utiliza para señalar la posición de un vehículo en el fotograma.

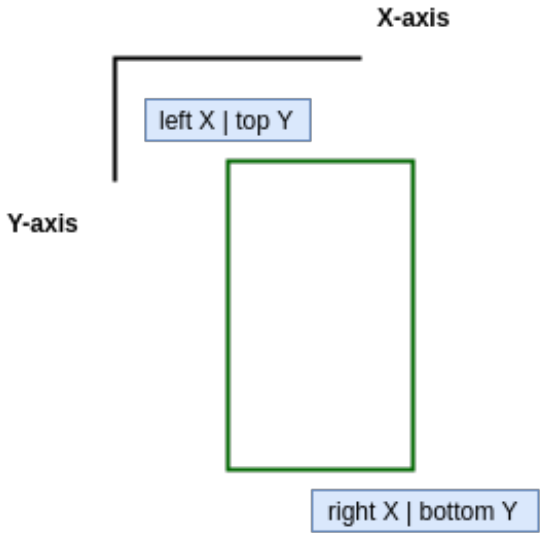

Figura 5.2: Representación de coordenadas de cuadro delimitador

Este etiquetado es necesario para entrenar nuestro propio modelo clasificador. La red de YOLOv3 utiliza estos datos, imagen y etiqueta, para aprender las características de cada objeto que nosotros requerimos.

En la Figura 5.3 se muestra la interfaz del programa LabelImg. En ella se puede observar la imagen que se está etiquetando, así como el cuadro delimitador que se dibujó de color verde. También aparece la lista de las clases que se han etiquetado hasta el momento.

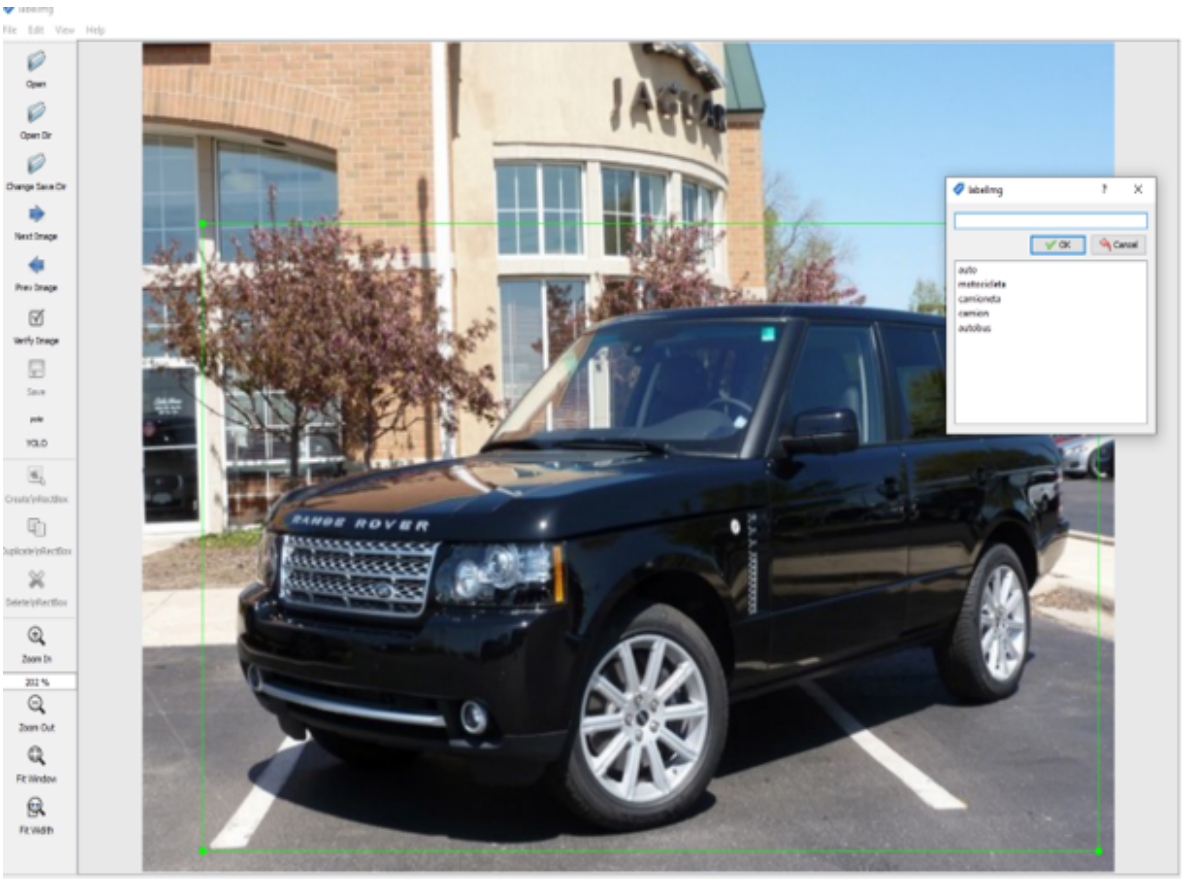

Figura 5.3: Interfaz de LabelImg

Como se puede observar en la Figura 5.4 el resultado del etiquetado de las imágenes es un archivo de texto con el mismo nombre de la imagen etiquetada, dentro de este archivo vienen la clase del vehículo y las coordenadas donde se identificó en la imagen. Todo esto en el formato necesario para el entrenamiento.

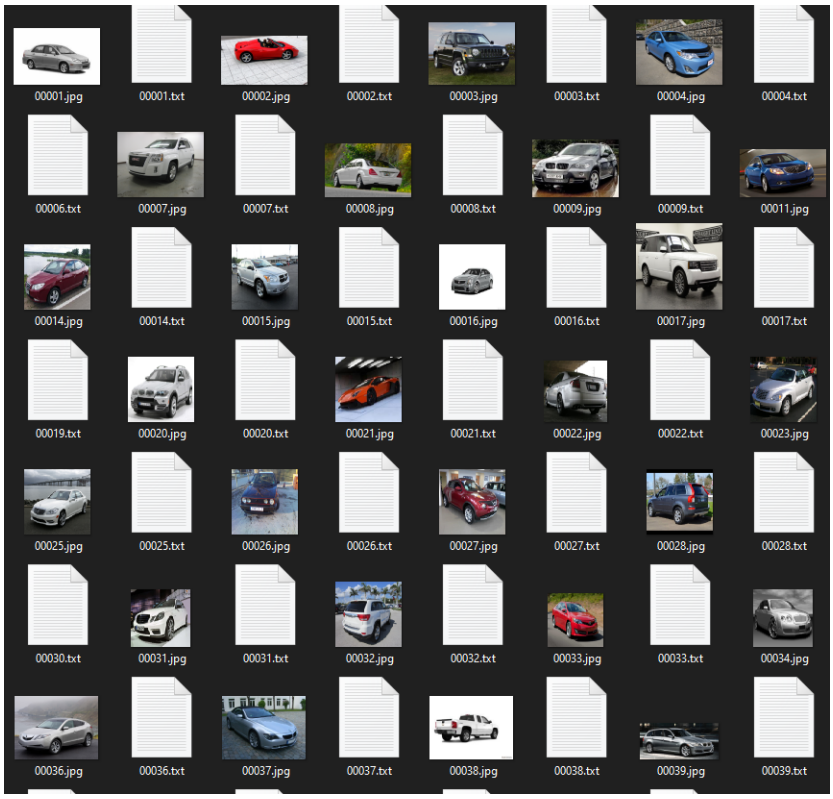

Figura 5.4: Muestra de imágenes de conjunto de datos junto a su etiqueta

5.2.3. Hardware utilizado

El equipo utilizado para llevar a cabo los entrenamientos del modelo predictivo de vehículos cuenta con las siguientes características:

- CPU: Intel i7 octava generación
- Memoria RAM: 32 GB 2666 MHz
- Tarjeta gráfica (GPU) 1: RTX 2080ti 11 GB
- Tarjeta gráfica (GPU) 2: RTX 2080ti 11 GB

Como se puede observar, cuenta con un poder computacional equiparable a una computadora de gama media actualmente. Su combinación de dos tarjetas gráficas RTX 2080ti, le proporciona una alta capacidad para llevar a cabo tareas de altos requisitos de memoria de aprendizaje profundo.

5.3. Configuración y análisis de experimentos

En la Tabla 5.2 se muestran las distintas configuraciones que fueron utilizadas en los diferentes entrenamientos. Se muestran cinco configuraciones distintas, estas fueron las principales utilizadas. Además de la duración de dicho entrenamiento y el resultado de la precisión promedio que se obtuvo. Esta precisión promedio se obtiene de las pruebas que hace la propia *DarkNet* al finalizar el entrenamiento. Como se puede observar, mientras se realizaban más entrenamientos, la precisión promedio aumentaba, esto gracias a la modificación de parámetros que se realizaba en cada nuevo entrenamiento. La modificación se realizó tomando en cuenta los resultados de entrenamientos anteriores. Así como la precisión promedio aumento, también lo hizo el tiempo de entrenamiento, alcanzando las 36 horas, esto debido al aumento en la cantidad de imágenes y su tamaño, así como la cantidad de épocas establecidas. Los principales parámetros que hacen que el tiempo de procesamiento aumente son: tamaño de imagen y *Batch*. El tamaño de imagen refiere a la cantidad de píxeles a la que la imagen se redimensiona. El Batch es la cantidad de imágenes, en un lote, que se procesan a al mismo tiempo por el equipo computacional. Entre más altos sean estos números, mayor es la cantidad de características que se pueden aprender. Es por esta razón que el tiempo de procesamiento aumenta. En cada entrenamiento la distribución de las imágenes era: 70 % para entrenamiento, 30 % para pruebas.

Tabla 5.2: Resultados obtenidos en entrenamientos con distintos parámetros

Parámetros	Entrenamiento 1	Entrenamiento 2	Entrenamiento 3	Entrenamiento 4	Entrenamiento 5
Imágenes por clase	500	1000	2000	2000	3000
Tamaño de imágenes	416x416	416x416	416x416	416x416	768x768
Batch	16	32	32	64	64
Épocas	14000	14000	14000	14000	18000
Duración	8 horas	20 horas	8 horas	20 horas	36 horas
Precisión Promedio	25.62 %	45.5 %	49.87 %	59.45 %	90.62 %

En la Tabla 5.3 se muestra los resultados de la precisión de la clasificación realizada por los modelos predictivos resultantes de cada entrenamiento. Así como en la Tabla 5.2, aquí también se observa el aumento de la precisión a medida que avanzan los entrenamientos,

alcanzando una mayor precisión en el último entrenamiento, siendo la clase Van la que obtuvo una mejor precisión con un 95.43 % y la clase Auto la que obtuvo el resultado menor con un 84.16 %.

Tabla 5.3: Precisión de clasificación obtenida por clase

Clase	Entrenamiento 1	Entrenamiento 2	Entrenamiento 3	Entrenamiento 4	Entrenamiento 5
Auto	12.07 %	28.16 %	40.47 %	54.45 %	84.16 %
Camioneta	21.03 %	56.62 %	49.05 %	65.54 %	93.22 %
Autobús	10.02 %	33.50 %	78.57 %	76.45 %	93.22 %
Van	8.62 %	38.34 %	57 %	65.75 %	95.43 %
Motocicleta	11.06 %	36.90 %	78.74 %	73.92 %	92.39 %
Bicicleta	42.43 %	55.64 %	53.18 %	58.44 %	87.92 %
Persona	9.02 %	16.14 %	22.08 %	21.58 %	85.09 %

En la Figura 5.5 se observa el comportamiento de la precisión de clasificación de cada clase durante los cinco entrenamientos mostrados. Es más fácil observar que durante los primeros entrenamientos la precisión del modelo era baja, incluso en el cuarto entrenamiento la precisión en algunas clases superaba apenas el 70 %. También se observa que la modificación de parámetros en los entrenamientos hace que el resultado en la precisión aumente. Estos datos se tomaron de la Tabla 5.3.

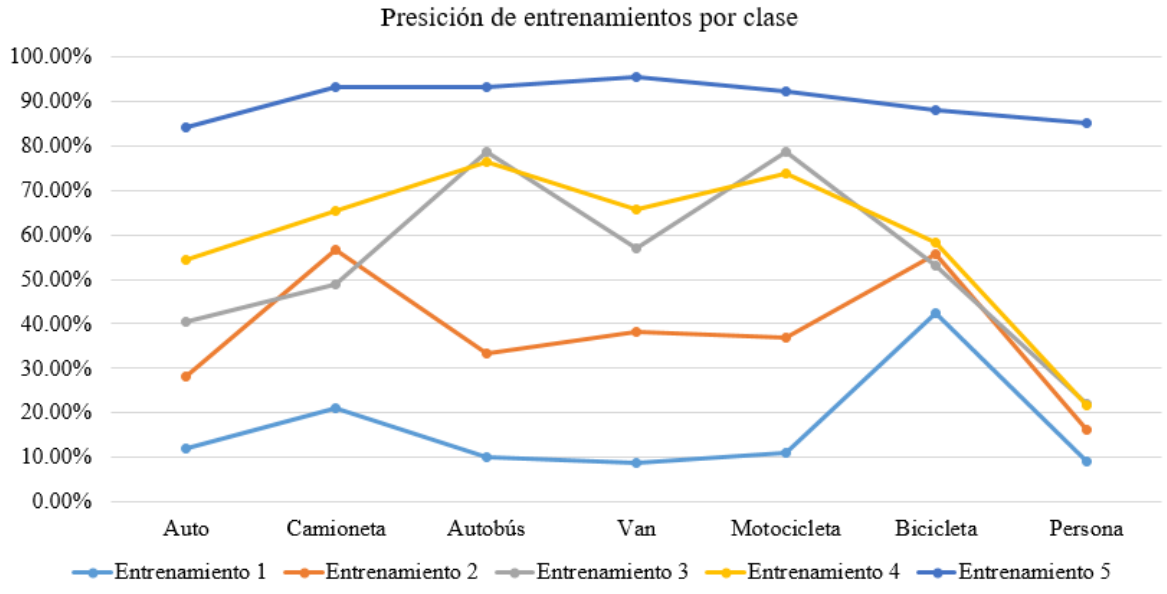

Figura 5.5: Gráfica de resultados de entrenamientos

En la Tabla 5.4 se muestran características claves que utilizan otros trabajos para llevar a

cabo la tarea de detección y seguimiento de objetivos. También se muestra el rendimiento que obtuvieron en la tarea principal en la que se enfocaron. Algunos de estos trabajos no llevan a cabo seguimiento, dado que se enfocan solo en la detección y clasificación. La detección de vehículos se ha dividido en tres métodos: (1) Método de Detección, (2) Objeto detectado y (3) Método de seguimiento, ya que podemos realizar este proceso con combinaciones de algoritmos enfocados a tareas específicas. Se destaca que cada trabajo presenta su forma de utilizar las características extraídas impactando en el rendimiento.

Tabla 5.4: Comparación de resultados entre trabajos del estado del arte

Autor	**Detección de vehículos**			**Rendimiento**
	Método de detección	**Objeto detectado**	**Método de seguimiento**	
Balid	Sensores Magnometricos	Vehículo	No hace seguimiento	99.98 % detección 97 % clasificación 97.11 % velocidad
Quan	Dos sensores geomagnéticos	Vehículo	No hace seguimiento	93 % velocidad
George	Sensor acústico	Vehículo	No hace seguimiento	73.42 % clasificación
Zhao	Sensor de vibración	Vehículo	No hace seguimiento	89 % clasificación
Huang	Sensores de rejilla de fibra de vidrio	Vehículo	No hace seguimiento	98.5 % clasificación
Tang	Haarcade AdaBoost	Vehículos	No hace seguimiento	97 % detección 91 % clasificación
Mandal	Segmentación	Colas de tráfico Vehículos estacionados	No hace seguimiento	92 % detección
Anil Rao	Sustracción de fondo	Vehículos	Filtro de Kalman Algoritmo Húngaro	+- 3km/h
Luvizon	SnooperText	Caracteres de placa vehícular	KLT SIFT	-3.24 km/h +3.91km/h
Zhang	Redes neuronales	Vehículos	No hace seguimiento	MSE 6.8 velocidad
Kampelmuhler	OpenCV	Vehículos	MediaFlow MIL	4.032 km/h
Niu	HaarCade	Vehículos	Comparación de distancia recorrida en cada fotograma	90 %
Zhao	Sensores LiDAR Redes neuronales	Zonas	Filtro de Kalman	95 % detección y seguimiento

Capítulo 6

Conclusiones

En esta sección se presentan las conclusiones hechas con base en la metodología propuesta en la sección 4 y los resultados que se mostraron en la sección 5. También se presentan las aportaciones que se hacen en el campo de la vigilancia de tráfico y los trabajos a futuro que se podrían realizar para llevar a cabo una mejora en el sistema.

6.1. Conclusión

Con la implementación de la metodología propuesta en este proyecto, se logró desarrollar un sistema que implementa técnicas de visión artificial y aprendizaje profundo para la extracción de características de tráfico vial. Algunas características que logra extraer de las imágenes son: clasificación vehicular, distancia recorrida entre dos puntos, tiempo que tarda en recorrer la distancia y frecuencia vehicular. Esto se hace a partir de un video tomado con una cámara por un lado de la carretera en situaciones no controladas, permitiendo tomar muestras en distintos puntos y áreas de manera sencilla. También ofrece el manejo de obstrucción y reaparición de vehículos. Durante la construcción del modelo clasificador se construyo un conjunto de datos de vehículos que cuenta con la cantidad de 21000 imágenes, estas se encuentran divididas en siete clases: Auto, Autobús, Camioneta, Motocicleta, Van, Bicicleta y Persona. Cada una de las imágenes cuenta con una etiqueta para llevar a cabo el entrenamiento.

Nuestro sistema alcanza una precisión en la detección y clasificación del 95.43 %. En comparación, otros trabajos como Balid et al., 2018 alcanzan una precisión de 99.98 % y

97 % en la detección y clasificación respectivamente. Con la diferencia que nuestro trabajo realiza una clasificación en siete clases distintas de vehículos y dicho trabajo clasifica por tamaño en tres distintos. Haciendo comparaciones similares con otros trabajos, sucede algo similar. Se observó que con los parámetros obtenidos y precisión del sistema, estos resultados son competitivos con los demás trabajos del estado del arte. Una ventaja de las ventajas de nuestro sistema, en comparación con otros trabajos que utilizan cámaras de video, es que no requerimos de una calibración previa para extraer características. Algunos trabajos como Anil Rao et al., 2015 utilizan transformación de espacios, lo que requiere un procesamiento y trabajo previo en cada instalación.

Como se vio en la toma de muestras, el equipo utilizado para obtener los videos puede instalarse por un lado de la carretera, permitiendo posicionar este equipo en distintas áreas o utilizar cámaras ya instaladas. Esto ofrece la posibilidad de no realizar una inversión en equipos especializados como: sensores, radares o cámaras RGB o 3D.

Con esto se demuestra que si es posible construir un sistema que utilice técnicas de visión artificial y redes neuronales convolucionales capaz de características de información vial a partir de secuencias de imágenes de tráfico, en situaciones no controladas. Se logró almacenar estas características en un archivo CSV, al cual se accede al finalizar el análisis del video. Este archivo puede ser utilizado por los departamentos de tránsito para llevar a cabo el análisis de tráfico vehicular de la zona.

6.2. Aportaciones

Gracias a este trabajo se construyo un sistema que es capaz de extraer distintos datos de los videos de tráfico vial utilizando equipos no especializados y que no requiere de una inversión de dinero alta. Los datos más relevantes que se extraen son: clasificación, conteo, distancia recorrida y tiempo en recorrer la distancia. Gracias a estos datos se pueden llevar a cabo análisis viales con el fin de mejorar las infraestructuras de las ciudades, la seguridad de la población, así como reducir el impacto que tienen los vehículos al emitir gases nocivos en el medio ambiente. Esto último debido a la disminución del uso de los vehículos por darle un mejor flujo al tráfico. Se creó un corpus de distintas clases de vehículos, el cual cuenta

con alrededor de, 21000 imágenes en total, todas ellas con su etiqueta de clasificación. Con este corpus se obtuvieron modelos de clasificación que junto con las redes convolucionales de YOLOv3 son capaces de hacer detección y clasificación de las siguientes clases: Auto, Autobús, Camioneta, Van, Motocicleta, Bicicleta y Persona. Se hizo la unió de distintos métodos, técnicas, herramientas y librerías de código abierto, que se utilizaron para crear el extractor de características.

6.3. Trabajos a futuro

Como trabajos a futuro a considerar para fortalecer algunas funciones, se puede tomar en cuenta la posición de la cámara en distintos puntos de la carretera. Esto evitaría la perdida de información cuando los vehículos son obstruidos. La mayoría de trabajos que utilizan cámaras necesitan realizar calibraciones en la cámara cuando se instala en algún lugar. Por lo que algo a buscar es que esta calibración se haga de manera automática, permitiendo que cualquier persona sea capaz de instalar el sistema, además de adaptarse fácilmente a cualquier lugar.

Referencias

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. (2016). {TensorFlow}: A System for {Large-Scale} Machine Learning. 12th USENIX symposium on operating systems design and implementation (OSDI 16), 265-283 (page 36).

Anderson, C. (2015). Docker [Software engineering]. IEEE Software, 32(3), 102-c3. https://doi.org/10.1109/MS.2015.62 (page 35)

Anil Rao, Y., Kumar, N. S., Amaresh, H. & Chirag, H. (2015). Real-time speed estimation of vehicles from uncalibrated view-independent traffic cameras. TENCON 2015 - 2015 IEEE Region 10 C 1-6. https://doi.org/10.1109/TENCON.2015.7373162 (pages 41, 69)

Balid, W., Tafish, H. & Refai, H. H. (2018). Intelligent Vehicle Counting and Classification Sensor for Real-Time Traffic Surveillance. IEEE Transactions on Intelligent Transportation Systems, 19(6), 1784-1794. https://doi.org/10.1109/TITS.2017.2741507 (pages 1, 38, 68)

Bochinski, E., Eiselein, V. & Sikora, T. (2017). High-speed tracking-by-detection without using image information. 2017 14th IEEE international conference on advanced video and signal based 1-6 (page 41).

Burnett, K., Samavi, S., Waslander, S. L., Barfoot, T. D. & Schoellig, A. P. (2019). aUTo-Track: A Lightweight Object Detection and Tracking System for the SAE AutoDrive Challenge. University of Toronto, 209-216. https://doi.org/10.1109/CRV.2019.00036 (pages 6, 37)

Coifman, B. & Neelisetty, S. (2014). Improved speed estimation from single-loop detectors with high truck flow. Journal of Intelligent Transportation Systems, 18(2), 138-148 (page 1).

Fernández Llorca, D., Hernández Martıénez, A. & Garcıéa Daza, I. (2021). Vision-based vehicle speed estimation: A survey. IET Intelligent Transport Systems (page 3).

Geiger, A., Lenz, P., Stiller, C. & Urtasun, R. (2013). Vision meets robotics: The kitti dataset. The International Journal of Robotics Research, 32(11), 1231-1237 (page 42).

George, J., Mary, L. & S, R. K. (2013). Vehicle detection and classification from acoustic signal using ANN and KNN. 2013 International Conference on Control Communication and Computing (I 436-439. https://doi.org/10.1109/ICCC.2013.6731694 (page 39)

Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. https://doi.org/10.2200/S00762ED1V01Y201703HLT037. (Page 20)

Huang, Y., Lu, P., Tolliver, D. et al. (2018). Vehicle classification system using in-pavement fiber Bragg grating sensors. IEEE Sensors Journal, 18(7), 2807-2815 (page 40).

INEGI. (2020). Accidentes de tránsito. https://www.inegi.org.mx/temas/accidentes/ (page 4)

Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. https://doi.org/10.48550/ARXIV.1502.03167. (Page 29)

Kampelmühler, M., Müller, M. G. & Feichtenhofer, C. (2018). Camera-based vehicle velocity estimation from monocular video. https://doi.org/10.48550/ARXIV.1802.07094. (Page 42)

Kassem, N., Kosba, A. E. & Youssef, M. (2012). RF-Based Vehicle Detection and Speed Estimation. 2012 IEEE 75th Vehicular Technology Conference (VTC Spring), 1-5. https://doi.org/10.1109/VETECS.2012.6240184 (page 2)

Kim, P. (2017). Convolutional Neural Network. MATLAB Deep Learning: With Machine Learning, Neural Net (pp. 121-147). Apress. https://doi.org/10.1007/978-1-4842-2845-6_6. (Page 26)

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (page 22).

Lee, H. & Coifman, B. (2015). Using LIDAR to Validate the Performance of Vehicle Classification Stations. Journal of Intelligent Transportation Systems, 19(4), 355-369. https://doi.org/10.1080/15472450.2014.941750 (page 2)

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L. & Dollár, P. (2014). Microsoft COCO: Common Objects in Context. https://doi.org/10.48550/ARXIV.1405.0312. (Page 37)

Liu, H., Ma, J., Xu, T., Yan, W., Ma, L. & Zhang, X. (2020). Vehicle Detection and Classification Using Distributed Fiber Optic Acoustic Sensing. IEEE Transactions on Vehicular Technology, 69(2), 1363-1374. https://doi.org/10.1109/TVT.2019.2962334 (page 2)

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of compute 60(2), 91-110 (page 42).

Luvizon, D. C., Nassu, B. T. & Minetto, R. (2014). Vehicle speed estimation by license plate detection and tracking. 2014 IEEE International Conference on Acoustics, Speech and Signal Processin 6563-6567. https://doi.org/10.1109/ICASSP.2014.6854869 (pages 4, 41)

Ma, W. & Lu, J. (2017). An equivalence of fully connected layer and convolutional layer. arXiv preprint arXiv:1712.01252 (pages 26, 27).

Mandal, V., Mussah, A. R., Jin, P. & Adu-Gyamfi, Y. (2020). Artificial intelligence-enabled traffic monitoring system. Sustainability, 12(21), 9177 (page 41).

Mcculloch, W. & Pitts, W. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 5 (page 10).

Min, Chen, Q., Lin & Yan, S. (2013). Network In Network. https://doi.org/10.48550/ARXIV. 1312.4400. (Page 29)

Minsky, M. & Papert, S. (1969). Perceptrons: An Introduction to Computational Geometry. MIT Press. (Page 11).

Niu, H., Gonzalez-Prelcic, N. & Heath, R. W. (2018). A UAV-Based Traffic Monitoring System - Invited Paper. 2018 IEEE 87th Vehicular Technology Conference (VTC Spring), 1-5. https://doi.org/10.1109/VTCSpring.2018.8417546 (pages 2, 43)

Overview. (s.f.). https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/overview. html. (Pages 35, 36)

Patterson, J. & Gibson, A. (2017). Deep learning: A practitioner's approach. .º'Reilly Media, Inc." (Pages 12, 17-19, 22-24).

Quan, W., Wang, H. & Gai, Z. (2020). Spot vehicle speed detection method based on short-pitch dual-node geomagnetic detector. Measurement, 158, 107661. https://doi.org/ https://doi.org/10.1016/j.measurement.2020.107661 (pages 2, 39)

Redmon, J. & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv (pages 5, 30, 31).

Rosebrock, D. A. (2017). Deep Learning for Computer Vision with Python. PYIMAGE SEARCH. (Page 18).

Rossum, G. V. (1994). Welcome to Python.org. https://www.python.org/shell/. (Page 34)

Rumelhart, D. E. & McClelland, J. L. (1986). Parallel Distributed Processing: Explorations in the Microstructu MIT Press. (Page 14).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014a). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Researc 15(56), 1929-1958. http://jmlr.org/papers/v15/srivastava14a.html (page 28)

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014b). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning researc 15(1), 1929-1958 (page 28).

Suhr, J. K. (2009). Kanade-lucas-tomasi (klt) feature tracker. Computer Vision (EEE6503), 9-18 (page 42).

Swamynathan, M. (2017). Mastering Machine Learning with Python in Six Steps A Practical Implementation Springer Science+Business Media New York. (Page 13).

Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science & Business Media. (Page 14).

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. & Liu, C. (2018). A survey on deep transfer learning. International conference on artificial neural networks, 270-279 (page 15).

Tang, Y., Zhang, C., Gu, R., Li, P. & Yang, B. (2017). Vehicle detection and recognition for intelligent traffic surveillance system. Multimedia tools and applications, 76(4), 5817-5832 (page 40).

Tzutalin. (2015). LabelImg. https://github.com/tzutalin/labelImg. (Page 61)

Ukani, V., Garg, S., Patel, C. & Tank, H. (2016). Efficient vehicle detection and classification for traffic surveillance system. International Conference on Advances in Computing and Data Sciences, 495-503 (page 40).

Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. (2021). Scaled-YOLOv4: Scaling Cross Stage Partial Network. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recogn 13029-13038 (page 57).

Welch, G., Bishop, G. et al. (1995). An introduction to the Kalman filter (pages 31, 33).

Won, M. (2020). Intelligent Traffic Monitoring Systems for Vehicle Classification: A Survey. IEEE Access, 8, 73340-73358. https://doi.org/10.1109/ACCESS.2020.2987634 (page 1)

Won, M., Zhang, S. & Son, S. H. (2017). WiTraffic: Low-Cost and Non-Intrusive Traffic Monitoring System Using WiFi. 2017 26th International Conference on Computer Communication and Ne 1-9. https://doi.org/10.1109/ICCCN.2017.8038380 (page 2)

Zaniolo, L. & Marques, O. (2020). On the use of variable stride in convolutional neural networks. Multimedia Tools and Applications, 79(19), 13581-13598 (page 23).

Zhang, Y., Wan, B., Liu, W. et al. (2017). Vehicle motion detection using CNN. IEEE Access, 5, 24023-24031 (page 42).

Zhao, H., Wu, D., Zeng, M. & Zhong, S. (2018). A vibration-based vehicle classification system using distributed optical sensing technology. Transportation Research Record, 2672(43), 12-23 (page 39).

Zhao, J., Xu, H., Liu, H., Wu, J., Zheng, Y. & Wu, D. (2019). Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors. Transportation Research Part C: Emerging Technolog 100, 68-87. https://doi.org/https://doi.org/10.1016/j.trc.2019.01.007 (page 43)