

SEP

TNM

INSTITUTO TECNOLÓGICO DE CULIACÁN



ESTIMACIÓN DE LA VELOCIDAD VEHICULAR A PARTIR DE
SECUENCIAS DE IMÁGENES MEDIANTE TÉCNICAS DE VISIÓN
ARTIFICIAL Y APRENDIZAJE MÁQUINA.

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:

ING. RAFAEL IMPERIAL ROJO
INGENIERO EN SISTEMAS COMPUTACIONALES

DIRECTOR DE TESIS:
DR. HÉCTOR RODRÍGUEZ RANGEL

CULIACÁN, SINALOA

Noviembre del 2021



Instituto Tecnológico de Culiacán

"2021, Año de la Independencia"

Culiacán, Sin., 19 de Noviembre del 2021

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
OFICIO: DEPI/275/11/2021

ASUNTO: Autorización Impresión

RAFAEL IMPERIAL ROJO
ESTUDIANTE DE LA MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
PRESENTE.

Por medio de la presente y en virtud de que ha completado los requisitos para el examen de grado de **Maestro en Ciencias de la Computación**, se concede autorización para la impresión de la tesis titulada: **"ESTIMACIÓN DE LA VELOCIDAD VEHICULAR A PARTIR DE SECUENCIAS DE IMÁGENES MEDIANTE TÉCNICAS DE VISIÓN ARTIFICIAL Y APRENDIZAJE MÁQUINA"** bajo la dirección del(a) **DR. HÉCTOR RODRÍGUEZ RANGEL**.

Sin otro particular reciba un cordial saludo.

ATENTAMENTE
Excelencia en Educación Tecnológica®

M.C. MARÍA ARACELY MARTÍNEZ AMAYA
JEFE(A) DE LA DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CULIACÁN
DEPARTAMENTO DE DIVISIÓN DE
ESTUDIOS DE POSGRADO E INVESTIGACIÓN

C.c.p. archivo

MAMA/lucy *



Juan de Dios Bátiz 310 Pte.
Col. Guadalupe, C.P. 80220
Culiacán, Sinaloa
Tel. 667-713-3804
tecnm.mx | culiacan.tecnm.mx





“ESTIMACIÓN DE LA VELOCIDAD VEHICULAR A PARTIR DE SECUENCIAS DE IMÁGENES MEDIANTE TÉCNICAS DE VISIÓN ARTIFICIAL Y APRENDIZAJE MÁQUINA”

Tesis presentada por el(a):

RAFAEL IMPERIAL ROJO

Aprobada en contenido y estilo por:

DR. HÉCTOR RODRÍGUEZ RANGEL
Director de Tesis

M.C. Gloria Ekaterine Peralta Peñúñuri.
Secretario

Dra. María Lucía Barrón Estrada
Vocal -1

M.C. Ricardo Rafael Quintero Meza
Vocal -2

M.C. María Aracely Martínez Amaya
Jefe(a) de la División de Estudios de
Posgrado e Investigación



Juan de Dios Bátiz 310 Pte.
Col. Guadalupe, C.P. 80220
Culiacán, Sinaloa
Tel. 667-713-3804
tecnm.mx | culiacan.tecnm.mx



Dedicatoria

Dedico este trabajo a mis padres Rosa Haydee Rojo Acosta y Rafael Imperial Sánchez quienes me han apoyado a lo largo de mi vida. Gracias a esto he podido plantearme metas a largo plazo, siendo un logro más alcanzar el nivel de maestría en mi carrera profesional, este logro también es de ellos. Se han esforzado en apoyarme, aconsejarme y dar lo mejor de ellos para mi crecimiento personal.

Agradecimientos

Agradezco a **mi familia** por su apoyo y motivación durante mis estudios.

Al **Instituto Tecnológico de Culiacán** por mi formación profesional en licenciatura y maestría.

Al **Consejo Nacional de Ciencia y Tecnología** por financiar mis estudios de maestría.

A **mis compañeros** por su apoyo y volverse mis amigos en el poco tiempo que hemos estado conviviendo como compañeros.

Al **Dr. Héctor Rodríguez Rangel** por su apoyo, su paciencia, sus aportes y compartir conmigo sus experiencias para que yo pueda alcanzar esta meta.

Al **profesorado y personal de posgrado** compartir sus conocimientos, su apoyo y su gestión del departamento de Posgrado.

Por último, a la **Dra. María Lucía Barrón Estrada** sin su apoyo y confianza no habría podido ingresar a esta Maestría, su labor como profesora para mí ha sido la más importante en mi carrera al enseñarme las bases de todo lo que se hoy en día, gracias a ella fue que nació en mi la pasión por la informática.

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.

Rafael Imperial Rojo. Culiacán, Sinaloa, México, 2021.

Resumen

La movilidad urbana se incrementó con el uso de automóviles, lo que originó también un aumento de accidentes de tránsito. Para estudiar este fenómeno se requiere de estudios de tránsito mediante equipo especial. Con los avances tecnológicos, la inteligencia artificial y el uso de videos es posible realizarlos sin modificar en gran medida la infraestructura urbana. Para el diseño de soluciones basadas en inteligencia artificial es necesario generar bases de datos públicas que proporcionen videos confiables para la calibración y desarrollo de soluciones que permitan realizar estudios de tránsito de manera automatizada. En este trabajo se presenta un sistema para generar un conjunto de datos a partir de videos grabados en un punto de observación de una vía carretera, obteniendo un total de 532 datos para analizar, los cuales fueron separados por el carril en el que fue detectado el vehículo. Este conjunto de datos fue utilizado para realizar experimentos con métodos estadísticos de correlación. La implementación de estos modelos de aprendizaje máquina se realizó mediante el uso bibliotecas como Scikit-Learn y Tensor Flow. Los mejores resultados arrojaron un Error Absoluto Medio (MAE) de 1.872 K/H para el carril central y 2.128 K/H para el último carril, aceptable comparados con el radar de velocidad Bushnell el cual tiene una precisión de 1.609 K/H. Debido a la falta de datos no fue posible realizar pruebas en el primer carril.

Palabras clave

- Análisis de tránsito vehicular
- Aprendizaje máquina
- Aprendizaje profundo
- Inteligencia artificial
- Velocidad vehicular
- Visión artificial

Índice general

Índice de figuras	XI
Índice de tablas	XIII
1. Introducción	1
1.1. Definición del problema	3
1.2. Hipótesis	4
1.3. Objetivo	4
1.4. Objetivos específicos	4
1.5. Justificación	4
1.6. Estructura de la tesis	5
2. Marco Teórico	6
2.1. Visión artificial	6
2.1.1. Representación de imágenes	6
2.1.2. Filtro Kalman	8
2.2. Redes neuronales	10
2.3. Aprendizaje Máquina	10
2.3.1. Aprendizaje supervisado	12
2.3.1.1. Clasificación	13
2.3.1.2. Regresión	14
2.3.1.3. Regresión Lineal	14
2.3.1.4. Regresión Ridge	14
2.3.1.5. Regresión Lasso	15
2.3.1.6. Elastic Net	16
2.3.2. Sobreajuste y Subajuste	16
2.3.3. Métricas de evaluación	18
2.4. Aprendizaje profundo	19
2.4.1. Redes neuronales	19
2.4.2. Perceptrón	20
2.4.3. Redes Neuronales Multicapa	21
2.4.4. Propagación hacia atrás (Backpropagation)	21
2.4.5. Descenso del Gradiente	23
2.4.6. Tasa de aprendizaje	24
2.4.7. Funciones de activación	25

2.4.7.1.	Función lineal	25
2.4.7.2.	Sigmoidal	25
2.4.7.3.	Tangente hiperbólica	26
2.4.7.4.	Softmax	27
2.4.7.5.	ReLU	27
2.4.8.	Funciones de pérdida	28
2.4.9.	Optimizadores	28
2.5.	Tecnologías utilizadas	29
2.5.1.	Python	29
2.5.2.	Contenedores Docker	30
2.5.3.	Nvidia Docker	31
3. Estado del Arte		32
3.1.	Detección de acuerdo a simple cálculo con fórmula física	32
3.2.	Detección de acuerdo a región de interés	33
3.3.	Detección usando cámaras estéreo y placa del vehículo	34
3.4.	Detección de acuerdo a líneas carreteras	35
3.5.	Detección de acuerdo a centroide del vehículo	36
3.6.	Detección usando redes neuronales	36
3.7.	Resumen	40
4. Metodología		43
4.1.	Obtención de conjunto de datos	44
4.1.1.	Toma de muestras	45
4.1.2.	Limpieza de las muestras	46
4.2.	Obtención de velocidad	48
4.2.1.	Extracción de características	48
4.2.2.	Modelo predictivo	55
4.2.2.1.	Estimar velocidad basado en factor de correlación	55
4.2.2.2.	Estimar velocidad utilizando modelos de regresión	57
4.2.3.	Obtención de la velocidad	58
5. Resultados		59
5.1.	Lugar para toma de muestras	59
5.2.	Muestras obtenidas	60
5.3.	Hardware utilizado	62
5.4.	Experimentos	62
5.5.	Regresión lineal para cálculo de la velocidad	63
5.6.	Resultados Regresión lineal para cálculo de la velocidad	63
5.7.	Modelos de Regresión	64
5.7.1.	Carril central	64
5.7.2.	Último Carril	65
5.8.	Modelo Neuronal	65
5.8.1.	Carril central	66
5.8.2.	Último carril	66

5.9. Análisis de resultados	67
6. Conclusiones	68
6.1. Conclusión	68
6.2. Aportaciones	69
6.3. Trabajo futuro	70
Bibliografía	71

Índice de figuras

2.1.	Algoritmo filtro Kalman (Welch, Bishop et al., 1995).	9
2.2.	<i>Dataset CIFAR-10 (CIFAR-10 s.f.).</i>	13
2.3.	Gráfica de regresión lineal.	14
2.4.	Regresión Ridge con diferentes valores de α (Géron, 2019).	15
2.5.	Ejemplo de subajuste.	17
2.6.	Ejemplo de sobreajuste.	17
2.7.	Ejemplo de ideal.	18
2.8.	Problema XOR, no es linealmente separable.	20
2.9.	Representación básica de red neuronal multicapa.	21
2.10.	Propagación hacia adelante y propagación hacia atrás.	23
2.11.	Mínimo local y global.	23
2.12.	Tasa de aprendizaje.	24
2.13.	Gráfica de función lineal.	25
2.14.	Gráfica de función Sigmoidal.	26
2.15.	Gráfica de función hiperbólica.	26
2.16.	Gráfica de función ReLU.	27
2.17.	Representación de un contenedor (<i>What is a container?</i> s.f.).	30
2.18.	Arquitectura de Nvidia Docker (<i>Overview</i> s.f.).	31
4.1.	Proceso de obtención de la velocidad.	44
4.2.	Proceso de obtención de conjunto de datos.	44
4.3.	Radar de velocidad Bushnell(<i>Buy velocity speed gun and more</i> s.f.).	45
4.4.	Lugar donde se tomaron las muestras.	46
4.5.	Límites para lugar de las muestras.	47
4.6.	Proceso de obtención de velocidad.	48
4.7.	Detección de vehículos dentro de recuadros.	49
4.8.	Detección de vehículos en punto B, sus trayectorias y triangulación del generado con el punto A y B.	49
4.9.	Punto A con vehículo en recuadro blanco.	50
4.10.	Punto B con vehículo en recuadro blanco.	51
4.11.	Imagen resultado para cada línea de archivo csv.	52
4.12.	Imagen válida con vehículo entrando representando una línea del archivo csv.	53
4.13.	Imagen válida con vehículo saliendo representando una línea del archivo csv.	53
4.14.	Imagen inválida (parte izquierda) con vehículo entrando representando una línea del archivo csv.	54

4.15. Imagen inválida (parte derecha) con vehículo saliendo representando una línea del archivo csv.	54
4.16. Diagrama de flujo para la creación de modelo con la fórmula física de la velocidad.	56
4.17. Diagrama de flujo para definir un modelo predictivo de la velocidad.	57
5.1. Lugar donde fueron tomadas las muestras.	60
5.2. Muestra de primer carril.	61
5.3. Muestra de segundo carril.	61
5.4. Muestra de último carril.	62
5.5. Red neuronal implementada en Tensor Flow.	66

Índice de tablas

2.1.	Resoluciones de video	7
3.1.	Resumen estado del arte	42
4.1.	Características obtenidas por el sistema.	51
5.1.	Resultados usando regresión simple	64
5.2.	Resultados para carril central	65
5.3.	Resultados para último carril	65
5.4.	Resultados utilizando Tensor Flow carril central	66
5.5.	Resultados utilizando Tensor Flow último carril	67

Capítulo 1

Introducción

El aumento del uso de vehículos motorizados se debe al incremento de la población centrándose en las zonas urbanas y la necesidad de movilidad en la vida cotidiana. Como consecuencia se tiene un aumento de tráfico vehicular, lo cual conlleva a un mayor número de accidentes vehiculares e incremento de gases de efecto invernadero, afectando la salud de la población en general. El uso de automóviles es esencial en nuestra vida cotidiana. Según la organización *Association for Safe International Road Travel (ASIRT)*, cada año mueren más de 1.3 millones de personas en accidentes de tráfico. Además, entre 20 y 50 millones de personas resultan heridas o incapacitadas (Zaki et al., 2020).

En caso de accidente, la mayor responsabilidad recae en el conductor del automóvil (Velázquez Narváez et al., 2017). Entre los principales factores que provocan los accidentes de tránsito se encuentran el exceso de velocidad, la conducción distraída, obstáculos en carretera, mala señalización, estado de la infraestructura vial, e iluminación. Según datos del Instituto Nacional de Estadística e Información Geográfica (INEGI, 2011), entre 1997 y 2009, los accidentes en la región aumentaron un 72.7 % en zonas urbanas y rurales (Carro-Pérez & Ampudia-Rueda, 2019).

Un elemento importante al revisar la causa de accidentes viales es el exceso de velocidad, por este motivo los estudios de tránsito vehicular se enfocan en revisar las causas de este elemento. Estos estudios requieren de un sistema monitoreo de velocidad eficaz. Además, se requiere de sistemas de control que asistan al conductor al transitar por las calles, conocidos como *ADAS (Advanced Driver Assistance Systems)* (Carro-Pérez & Ampudia-Rueda, 2019).

Actualmente, para un estudio de tránsito se requiere de equipo especializado para generar

los datos necesarios para analizar la situación y obtener posibles soluciones; uno de estos dispositivos son los radares de velocidad los cuales utilizan ondas de radio para calcular el tiempo que le toma a la onda viajar hasta el vehículo y de vuelta al radar para determinar la velocidad con la que viaja el vehículo. A su vez estos dispositivos son utilizados en puntos estratégicos de la ciudad para monitorear la velocidad de los vehículos y captar una imagen del vehículo para elaborar una multa por exceso de velocidad.

Ahora bien, para un estudio de tránsito se requieren equipos especializados como los radares de velocidad, pero usar estos equipos no es tan viable por su elevado costo. Sin embargo, actualmente las ciudades cuentan con un sistema de videovigilancia, en donde es posible usar este equipo existente en la ciudad para determinar varias características del flujo vehicular como la velocidad, accidentes, entre otros.

La estimación del tráfico puede proporcionarse a los usuarios finales, a los departamentos de seguridad del estado, a los departamentos de planeación, etc. entre otros (Impedovo et al., 2019).

La inteligencia artificial está revolucionando las actividades cotidianas que lleva a cabo la sociedad moderna. Los investigadores y desarrolladores tanto de la industria automotriz como de seguridad vial están desarrollando activamente enfoques de conducción autónoma y monitoreo basados en el aprendizaje profundo. Para esto, es necesario contar con una red neuronal capaz de detectar vehículos capturados mediante video con la finalidad de determinar su velocidad, clasificarlos, detectar accidentes, por mencionar algunas tareas.

Rao & Frtunikj, 2018 describen las posibilidades y desafíos de integrar el aprendizaje profundo en vehículos autónomos en donde se estudia la creación de base de datos para el entrenamiento de dichas redes neuronales. En este trabajo se presenta una herramienta para determinar la velocidad vehicular a partir de secuencias de imágenes. Para el desarrollo de esta herramienta fue necesario la elaboración de un conjunto de datos experimental propio para después con técnicas de inteligencia artificial y estadísticas extraer la información necesaria para la estimación de la velocidad.

1.1. Definición del problema

Actualmente hay un incremento de la población en las grandes ciudades, entre algunas razones del crecimiento es que, estas tienden a proporcionar más trabajos. Este incremento de población ha provocado diferentes problemas entre ellos el de movilidad. En las ciudades grandes o pequeñas, los vehículos de motor son el principal medio con el cual la población es capaz de trasladarse ya sea para la recreación o para llegar a sus lugares de trabajo. El aumento de población y aumento de vehículos también son causantes de gran cantidad de emisiones de gases de efectos invernaderos, los cuales a su vez provocan problemas de salud para la población en general.

Un flujo de vehículos sin pausas prolongadas ayudan a disminuir los problemas de tráfico y la reducción de contaminación. Mejorar el flujo vehicular es importante para disminuir las consecuencias de la gran cantidad de vehículos. El análisis de tráfico se utilizan para identificar características como las horas pico, accidentes viales, la velocidad, el conteo de vehículos, entre otros. El presente trabajo está centrado en el desarrollo de una herramienta capaz de determinar la velocidad a la que viajan los vehículos a partir de secuencias de imágenes.

El enfoque tradicional para realizar estudios de tránsito está influenciado por el uso de equipo especial. Estos se pueden instalar bajo la superficie de la carretera tales como espires inductivas, sensores de campo magnético, contadores de ejes, sensores capacitivos y piezoelectrónicos. Por otro lado, se pueden instalar encima de la carretera, como los son los detectores de radares de microondas, detectores de radar láser, sensores de campo magnético, sensores infrarrojos pasivos y activos, sensores ultrasónicos, entre otros instrumentos; los cuales tienen un costo elevado además de necesitar una instalación especializada.

Las cámaras de videovigilancia son dispositivos instalados en puntos estratégicos de las grandes ciudades. Es por esto que se busca agregarles la capacidad de estimar la velocidad de los vehículos con la ayuda de inteligencia artificial. Buscando aprovechar la infraestructura existente y ahorrar en la adquisición de nuevos dispositivos especializados.

1.2. Hipótesis

Es posible determinar la velocidad de vehículos en movimiento a partir de secuencias de imágenes utilizando técnicas de inteligencia artificial y visión artificial.

1.3. Objetivo

Determinar la trayectoria y velocidad de un vehículo usando secuencias de imágenes a través de técnicas de inteligencia artificial y visión artificial.

1.4. Objetivos específicos

- Generar un conjunto de datos que permitan diseñar y entrenar un modelo capaz de terminar la velocidad de un vehículo a partir de secuencias de imágenes.
- Diseñar e implementar un modelo que permita la detección de un vehículo.
- Diseñar e implementar un algoritmo para determinar la trayectoria de un vehículo.
- Diseñar e implementar una herramienta para la determinación de la velocidad de vehículos mediante secuencias de imágenes.

1.5. Justificación

Con la implementación de un sistema para determinar la velocidad de vehículos en cámaras de videovigilancia, se proporciona a los centros de monitoreo datos para analizar e identificar los puntos más importantes en los que hay que poner especial atención para evitar posibles accidentes viales que podrían costar la pérdida de vidas o simplemente el retraso en los tiempos de traslados provocados por la congestión de tráfico lo cual a su vez ayuda a disminuir los gases de efecto invernadero nocivos para la salud.

Además, con la integración del sistema propuesto se espera un ahorro económico, puesto que no es necesario adquirir equipo especializado para determinar la velocidad de vehículos.

Por ejemplo, de radar de velocidad Bushnell tiene un costo promedio de 6,000 pesos mexicanos. Sin embargo, este dispositivo solo puede guardar la velocidad en pantalla, sin guardar evidencia del vehículo al que se le tomó la velocidad. Por otra parte, con la ayuda de cámaras de videovigilancia y este sistema es posible agregar la funcionalidad para determinar la velocidad de vehículos, aprovechando equipos existentes en la mayoría de las grandes ciudades, además de darle doble funcionalidad a estos equipos a los cuales no hay que realizar una instalación extra que provoquen pausar el tráfico, como en el caso de lazos inductivos, ya que el sistema no es invasivo con sistemas de vigilancia existentes.

1.6. 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 inteligencia artificial y visión artificial necesarios para comprender el contenido de este trabajo.
- Capítulo 3 - Estado del arte: Se presentan trabajos relacionados con determinar la velocidad de objetos.
- Capítulo 4 - Metodología: Se explica el proceso para el desarrollo del sistema, así como las herramientas utilizadas y la razón de utilizar ciertas herramientas.
- Capítulo 5 - Análisis de los resultados: Se presentan los resultados obtenidos utilizando el sistema y comparándolo con métodos convencionales como los radares de velocidad.
- Capítulo 6 - Conclusiones: Se presenta de manera resumida el proyecto, al mismo tiempo que el aporte del trabajo al área y el trabajo futuro que podría realizarse.

Capítulo 2

Marco Teórico

Este capítulo introduce a los conceptos más relevantes para el desarrollo del presente trabajo. Se abordan conceptos de Visión Artificial, Inteligencia Artificial, Aprendizaje Máquina, Aprendizaje Profundo, entre otros.

2.1. Visión artificial

La visión artificial intenta describir el mundo que vemos en imágenes y reconstruir sus propiedades (Szeliski, 2010), con ayuda de la visión artificial es posible darle la capacidad a una máquina para identificar, analizar y procesar imágenes del mundo real. En esta sección se explicará como, una máquina representa una imagen para después representarla en una pantalla o simplemente procesarla. Por otro lado se define el filtro Kalman el cual fue utilizado para el seguimiento de objetos.

2.1.1. Representación de imágenes

En computación una imagen se representa como una cuadrícula, donde cada recuadro es llamado píxel. Un píxel es la unidad más pequeña de la imagen y representa un color (Rosebrock, 2017). Con lo anterior se entiende, que para una imagen de 1000 x 600 se tienen, 1000 píxeles en cada fila y 600 píxeles para cada columna resultando 600,000 píxeles en total. Además, son dos las representaciones más comunes de un píxel (i.e escala de grises, color).

Cuando se representa una imagen utilizando escala de grises cada píxel tiene un valor numérico en el rango de 0 a 255. Con este rango de valores es posible variar la intensidad

de color presentado, siendo 0 el valor para representar el color negro y 255 el color blanco. Teniendo esto en cuenta se puede interpretar que los valores intermedios son diferentes intensidades de grises, de ahí el nombre escala de grises.

Para los píxeles a color existen diferentes espacios de colores, en este trabajo solo se hablara acerca del espacio de color RGB. Los píxeles RGB son representados usando 3 valores de intensidad para los colores rojo, verde y azul. Cada uno de estos 3 valores tiene un rango del 0 al 255 con lo cual se pueden variar los colores presentados. Al igual que la escala de grises cada uno de estos 3 valores pueden tener valores en el intervalo de 0 a 255. Al colocar los 3 valores de RGB a '0' (cero) se representa el color negro, por el contrario, para representar el color blanco los 3 valores de RGB deben ser de 255.

Ahora bien, para el caso de la representación de los videos, estos se pueden representar utilizando escala de grises o a color. Ya que los videos son una secuencia de imágenes con las cuales se genera la sensación de movimiento. Para el caso de los videos, al conjunto de imágenes que forman el video se les llama fotogramas y pueden variar en su frecuencia. A la frecuencia de imágenes contenidas en un tiempo determinado se le conoce como fotogramas por segundo. Hoy en día los fotogramas por segundo más comunes son 60, sin embargo, ya hay implementaciones con una mayor cantidad. La resolución tanto de una imagen como de un video se establece por la cantidad de píxeles en el eje X y el eje Y. Existen diferentes resoluciones de las cuales la más común es *Full HD*, en la Tabla 2.1 se muestra el resto de las resoluciones más comunes.

Tabla 2.1: Resoluciones de video

SIGLAS	NOMBRE	RESOLUCIÓN
SD	<i>Standard Definition</i>	640 x 480 píxeles
QHD	<i>Quarter of High Definition</i>	960 x 540 píxeles
HD	<i>High Definition</i>	1.280 x 720 píxeles
FHD	<i>Full HD o Full High Definition</i>	1.920 x 1.080 píxeles
QHD	<i>Quad High Definition</i>	2.560 x 1.440 píxeles
UHD	<i>Ultra High Definition</i>	3.840 x 2.160 píxeles

2.1.2. Filtro Kalman

El filtro Kalman es una solución recursiva al filtrado lineal de datos discretos (Welch, Bishop et al., 1995). Se encarga de estimar las variables de estado en un sistema dinámico, lo cual, en el caso del presente trabajo, implica estimar la siguiente ubicación de un vehículo identificado.

El filtro de Kalman tiene como objetivo resolver el problema general de estimar el estado $x \in \mathbb{R}^n$ de un proceso controlado en tiempo discreto, el cual se representa con la Ecuación 2.1.

La matriz A de dimensión $n \times n$ en la Ecuación 2.1 relaciona el estado en el paso de tiempo k con el estado en el paso $k + 1$, en ausencia de una función de excitación o ruido de proceso. La matriz B de dimensión $n \times 1$ relaciona la entrada de control $u \in \mathbb{R}^1$ con el estado x . La matriz H de dimensión $m \times n$ en la ecuación de medición (Ecuación 2.2) relaciona el estado con la medición z_k .

$$x_{k+1} = A_k x_k + B u_k + w_k \quad (2.1)$$

Filtro Kalman. Ecuación de estado

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

$$z_x = H_k x_k + v_k \quad (2.2)$$

Filtro Kalman. Relación estado y medición

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 2.1.

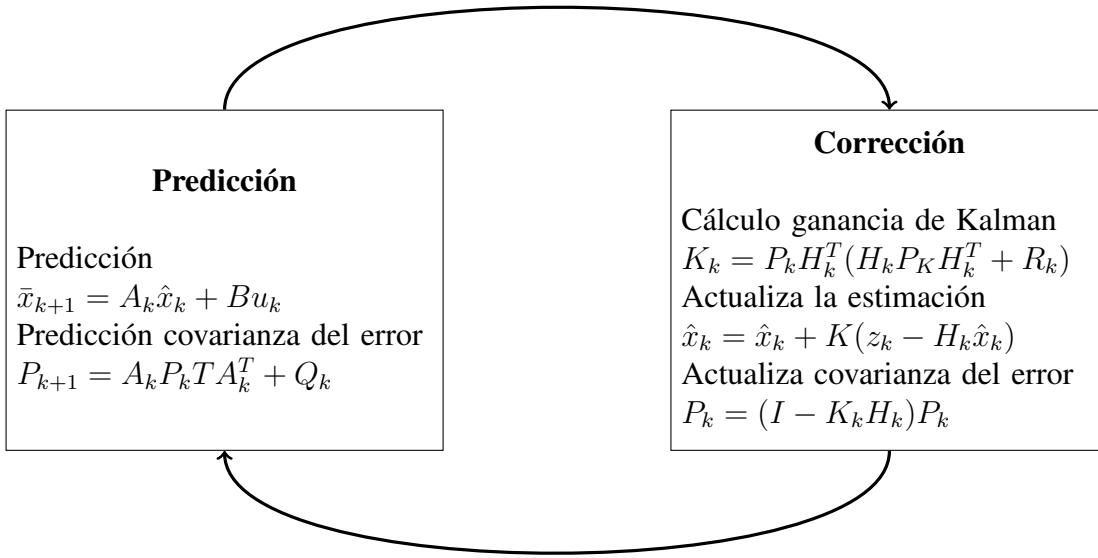


Figura 2.1: Algoritmo filtro Kalman (Welch, Bishop et al., 1995).

Para la fase de predicción se toman las Ecuaciones 2.3 y 2.4:

$$\bar{x}_{k+1} = A_k \hat{x}_k + Bu_k \quad (2.3)$$

$$P_{k+1} = A_k P_k T A_k^T + Q_k \quad (2.4)$$

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 2.5, 2.6 y 2.7:

$$K_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1} \quad (2.5)$$

$$\hat{x}_k = \hat{x}_k + K(z_k - H_k \hat{x}_k) \quad (2.6)$$

$$P_k = (I - K_k H_k) P_k \quad (2.7)$$

La Ecuación 2.5 representa el primer paso 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 2.6. El último paso es obtener una estimación de la covarianza del error a posteriori mediante la Ecuación 2.7.

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.

2.2. Redes neuronales

Las redes neuronales comenzaron a usarse en la década de 1940 con el artículo de Warren McCulloch y Walter Pitts publicado en 1943 (McCulloch & Pitts, 1943), con el cual demostraron que es posible calcular cualquier función aritmética o lógica con el uso de redes neuronales artificiales.

En 1950 Frank Rosenblatt (Rosenblatt, 1958) crea la red perceptrón, la cual es conocida como la primera aplicación práctica de las redes neuronales. Rosenblatt construyó una red perceptrón capaz de reconocer patrones. Esto dio inicio a la investigación del campo de las redes neuronales. No obstante, tiempo después Minsky and Papert (Minsky & Papert, 1969) demostraron que la red perceptrón solo podía resolver problemas específicos.

Con la publicación de Minsky and Papert muchos investigadores creyeron que no tenía sentido continuar con las redes neuronales. Además, de la limitante computacional de la época para realizar los experimentos que se requieran, llevando a una pausa para la investigación de las redes neuronales por una década.

En la década de 1980 David Rumelhart y James McClelland (Rumelhart & McClelland, 1986) crean el algoritmo de propagación hacia atrás para el entrenamiento de las redes perceptrón multicapa, lo cual marca el renacimiento de las redes neuronales. Este algoritmo fue la respuesta al libro de Minsky y Papert de 1960, en el que plantean la limitante del uso de perceptrón.

2.3. Aprendizaje Máquina

La inteligencia artificial incorpora un conjunto diverso de trabajos relacionados con el razonamiento automático mientras que el subcampo del aprendizaje máquina se especializa

en el reconocimiento de patrones y el aprendizaje de los datos (Rosebrock, 2017).

El aprendizaje máquina utiliza algoritmos para extraer información o patrones de un conjunto de datos (*Dataset*) sin procesar y representarla en un modelo, después usar este modelo para inferir resultados sobre otros datos que aún no han sido modelado (Patterson & Gibson, 2017).

Tradicionalmente se encuentran problemas donde la solución resuelve el problema todo el tiempo, sin embargo, existen problemas de los cuales no se sabe lo suficiente, por lo cual no es posible resolverlos de igual forma, es en este tipo de problemas, donde se utilizan las estadísticas para realizar estimaciones sobre la respuesta más cercana a la correcta al igual que lo hace el aprendizaje máquina (Harrington, 2012).

En otras palabras, el aprendizaje máquina es un subcampo de la inteligencia artificial, este se encarga de aprender los patrones con los cuales es posible modelar un conjunto de datos, una vez aprendidos los patrones más importantes se crea un modelo el cual es utilizado para realizar tareas de clasificación, regresión o pronóstico.

La metodología que se desea utilizar en el aprendizaje máquina es relativamente sencilla, por ejemplo, se desea aprender a identificar si una imagen contiene un animal como un ratón o un gato, se debe tomar cierto número de muestras (cientos o miles de preferencia). Con estas muestras el aprendizaje máquina se encargará de tomarlas y procesarlas para identificar las características más importantes. Estas características son el modelo que define a cada animal. Ahora bien, se puede tomar una de las muestras y usarla como entrada por el modelo creado, con esto es muy probable que la salida del modelo indique correctamente. Es por esto que se utilizan muestras que no hayan sido procesadas anteriormente.

La Real Academia Española dedicada a la regularización lingüística hispanohablante define aprender como adquirir el conocimiento de algo por medio del estudio o de la experiencia. En este caso la máquina aprende por medio de la experiencia adquirida al procesar cada una de las muestras. A tal grado que identifica muestras que no estaban en el conjunto de datos de entrada. En aprendizaje máquina se le llama entrenamiento al proceso de generación de un modelo a partir de las muestras. Mientras que la validación es el proceso con el cual el modelo infiere resultados a partir de muestras que no pasaron por el proceso de entrenamiento. El conjunto de datos son todas las muestras que serán utilizadas para el proceso de

entrenamiento y validación.

Comúnmente el conjunto de datos es dividido para cada proceso, entrenamiento y validación, en la práctica lo más común es separar el conjunto de un porcentaje de 70-30, siendo 70 % para el proceso de entrenamiento y 30 % para la validación.

Mas adelante se presentan las métricas que se utilizan para evaluar el modelo por medio del conjunto de datos de validación. En el aprendizaje máquina se tienen tres tipos diferentes de aprendizaje: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje semi-supervisado. A continuación, se describen los tres tipos de aprendizaje, sin embargo, en este trabajo solo se hablará con más detalle sobre el aprendizaje supervisado.

- Aprendizaje supervisado: Es un proceso de entrenamiento en el que se realizan predicciones sobre los datos de entrada y luego se corrigen las predicciones incorrectas. El proceso continúa hasta que se obtiene un error bajo o un número máximo de iteraciones. Para esto es necesario que los datos de entrada tengan cierta etiqueta con el significado de cada uno de los datos.
- Aprendizaje no supervisado: En este aprendizaje no se cuenta con la etiqueta correspondiente a su valor para cada dato. Para este caso es el entrenamiento el encargado de encontrar cierto patrón en los datos con el cual asigna una etiqueta a cada uno de ellos.
- Aprendizaje semi-supervisado: En el aprendizaje semi-supervisado se tienen los datos etiquetados y datos sin etiquetar. El proceso de entrenamiento toma los datos conocidos, los analiza y etiqueta cada uno de los datos no etiquetados para usarlos como datos de entrenamiento adicionales. El algoritmo semi-supervisado aprende la “estructura” de los datos.

2.3.1. Aprendizaje supervisado

En la Sección 2.3 se abordó el tema del aprendizaje supervisado, ahora se profundizará sobre los dos tipos de problemas que resuelve, el de clasificación y el de regresión.

2.3.1.1. Clasificación

En el aprendizaje máquina la clasificación consiste en que el algoritmo aprenda los patrones que definen un conjunto de datos perteneciente a una clase específica. Los valores de salida del modelo es la categoría correspondiente a los datos de entrada que pueden ser dos o más opciones bien definidas.

La clasificación binaria es la forma más simple de clasificación en esta se tienen solo dos valores de salida (0 o 1). Esta clasificación sirve para responder a la simple pregunta si pertenece o no a una clase en concreto. Por ejemplo, identificar si una transacción bancaria es fraudulenta o no.

Al revisar la literatura, se encuentran una gran variedad de conjuntos de datos con los cuales se puede resolver el problema de clasificación (i.e. MNIST, Animals, CIFAR-10, Flowers-17, entre otros). Donde, cada uno de estos conjuntos de datos sirven para evaluar tu modelo de aprendizaje propuesto.

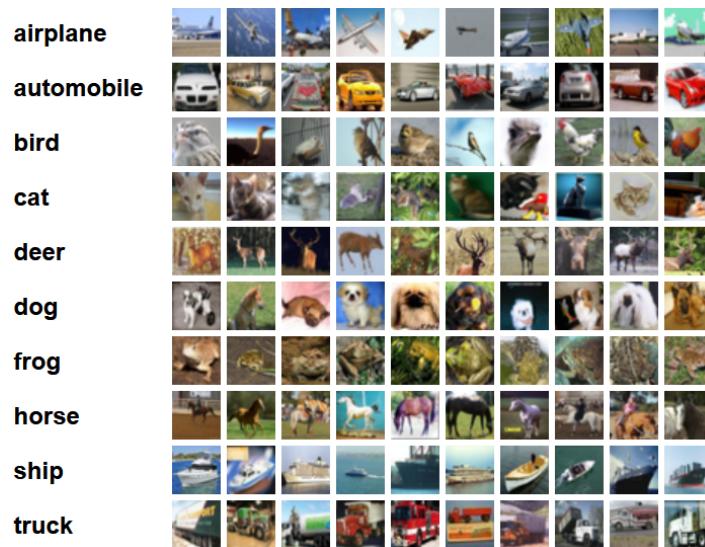


Figura 2.2: Dataset CIFAR-10 (CIFAR-10 s.f.).

Para todo aquel problema que implique separar un conjunto de datos en diferentes clases o categorías, para un número finito de clases, se utiliza la clasificación.

2.3.1.2. Regresión

La regresión es el proceso de volver hacia atrás en una actividad, para su análisis, con lo cual se busca identificar aquellas que más se repiten. Los problemas de regresión predicen un valor real. En otras palabras, la función que estima la variable dependiente conociendo la variable independiente.

La regresión es utilizada para aquellos problemas que implican predecir valores numéricos, un ejemplo de esto son las predicciones de ráfagas de vientos las cuales dando como entrada diferentes características del ambiente es posible predecir la velocidad del viento.

2.3.1.3. Regresión Lineal

La clase más común de regresión es la regresión lineal, ya que este tipo de regresión se puede modelar con una línea recta que modela la mayoría de los datos de muestreo. La regresión lineal intenta llegar a una función que describa la relación entre x y y , para valores conocidos de x , predice valores de y que resultan ser precisos (Patterson & Gibson, 2017).

La Figura 2.3 representa la regresión en una gráfica.

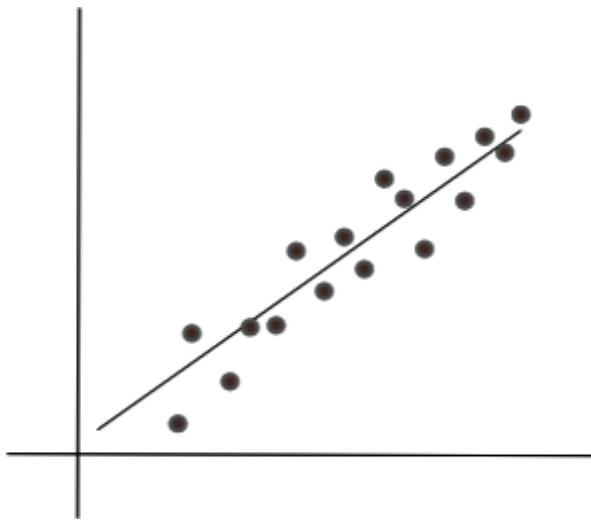


Figura 2.3: Gráfica de regresión lineal.

2.3.1.4. Regresión Ridge

La regresión Ridge es una versión regularizada de la regresión lineal, agrega un término de regularización en la función de costo (Ecuación 2.8). Con esto se busca que el algoritmo

mantenga los pesos lo más pequeños posible.

$$\alpha \sum_{i=1}^n (\theta_i)^2 \quad (2.8)$$

El hiperparametro α controla la regularización del modelo, si $\alpha = 0$ la regresión Ridge es simplemente una regresión lineal. Si α es muy grande, todos los pesos terminan igual a cero y resulta en una línea plana que pasa por la media de los datos(Figura 2.4).

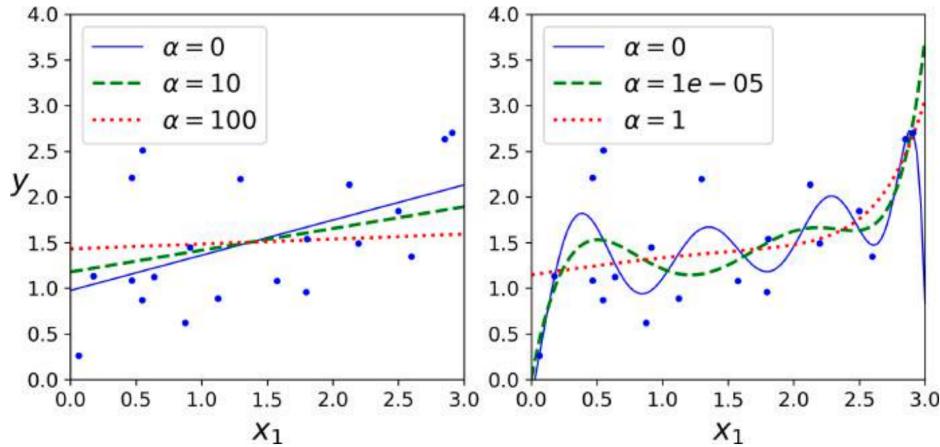


Figura 2.4: Regresión Ridge con diferentes valores de α (Géron, 2019).

La Ecuación 2.9 es la función de costo para la regresión Ridge.

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^n (\theta_i)^2 \quad (2.9)$$

2.3.1.5. Regresión Lasso

La regresión Lasso (*Least Absolute Shrinkage and Selection Operator* en Inglés) al igual que la regresión Ridge es una versión regularizada de la regresión lineal, sin embargo, la regresión Lasso utiliza valores absolutos en la función de regulación (Ecuación 2.10), una característica importante con Lasso es que puede llegar a eliminar por completo los pesos de las características menos importantes.

$$\alpha \sum_{i=1}^n | \theta_i | \quad (2.10)$$

La Ecuación 2.11 es la función de costo para la regresión Lasso.

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^n |\theta_i| \quad (2.11)$$

2.3.1.6. Elastic Net

Elastic Net es una versión regularizada de la regresión lineal, siendo el punto medio entre la regresión Ridge y Lasso, utiliza un valor de regularización r , cuando $r = 0$ Elastic Net es equivalente a Ridge, y cuando $r = 1$, se comporta como Lasso (Ecuación 2.12).

$$r\alpha \sum_{i=1}^n |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^n (\theta_i)^2 \quad (2.12)$$

La Ecuación 2.13 es la función de costo para la regresión Elastic Net.

$$J(\theta) = MSE(\theta) + r\alpha \sum_{i=1}^n |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^n (\theta_i)^2 \quad (2.13)$$

2.3.2. Sobreajuste y Subajuste

Cuando es entrenado un nuevo modelo se busca que tenga la capacidad de generalizar de tal manera que pueda interpretar datos de entrada que nunca haya visto.

Un buen modelo se equivoca poco en sus predicciones, esto significa que tiene un error bajo. Esto se evalúa ingresando al modelo datos que no recibió en su proceso de entrenamiento, de esta manera se tiene la seguridad de que el modelo es capaz de generalizar.

El subajuste ocurre cuando el modelo no puede obtener una pérdida suficientemente baja en el conjunto de entrenamiento. En este caso, el modelo no aprende los patrones en los datos de entrenamiento. Lo cual implica una gran cantidad de errores cuando es utilizado para realizar predicciones, como muestra la Figura 2.5.

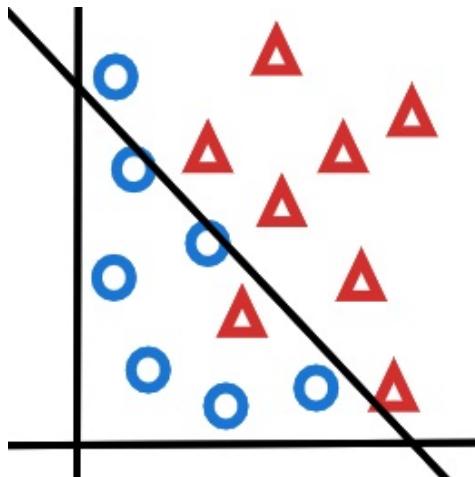


Figura 2.5: Ejemplo de subajuste.

Por otro lado, un sobreajuste es cuando la red modela los datos de entrenamiento demasiado bien y no se generaliza a los datos de validación (Rosebrock, 2017). En este caso, las predicciones solo serán buenas para aquellos datos que se utilicen en el mismo conjunto de datos que en el entrenamiento, Figura 2.6.

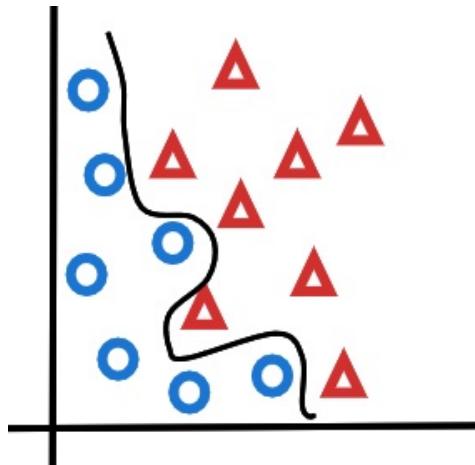


Figura 2.6: Ejemplo de sobreajuste.

Por otra parte, el entrenamiento ideal es cuando la red modela los datos de entrenamiento con cierto margen de error, pero siendo su gran mayoría correcta, como se muestra en la Figura 2.7.

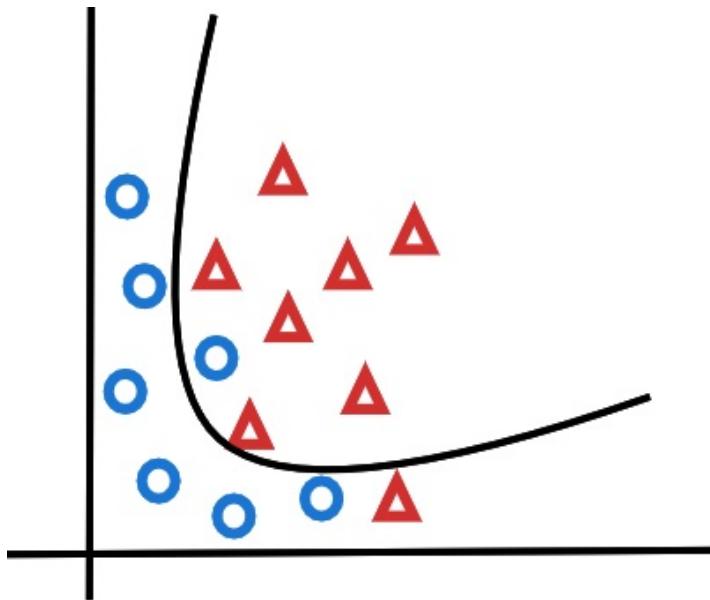


Figura 2.7: Ejemplo de ideal.

2.3.3. Métricas de evaluación

Ahora bien, cuando se realiza el entrenamiento es necesario saber que tan bueno es el modelo que se ha generado, para esto existen diferentes métricas que nos indican de manera cuantitativa la calidad del modelo.

Cuando se ha entrenado un modelo para un problema de clasificación se utilizan medidas de exactitud como una métrica de evaluación. La exactitud se calcula con el número de predicciones correctas entre el número de predicciones totales, La Ecuación 2.14 muestra la métrica de exactitud.

$$\text{Exactitud} = \frac{\text{Predicciones correctas}}{\text{Total de predicciones}} \quad (2.14)$$

Por otra parte, los modelos creados para problemas de regresión hay tres métricas principales MAE, MSE Y RMSE.

La métrica de evaluación del Error Absoluto Medio (MAE – *Mean Absolute Error* en Inglés) calcula la media de los valores absolutos de la diferencia entre los valores reales y los predichos. En otras palabras, MAE es el promedio de todos los errores sin tomar en cuenta si este error es positivo o negativo con respecto al valor real. La Ecuación 2.15 define el cálculo

de MAE.

$$MAE = \left(\frac{1}{n} \right) \sum_{i=0}^n |y_i - x_i| \quad (2.15)$$

El Error Medio Cuadrado (MSE – *Mean Square Error* en Inglés) se encarga de elevar al cuadrado el error, penalizando fuertemente los valores que se alejan demasiado del valor real esperado, con este valor se obtiene el promedio, la Ecuación 2.16 define el calculo de MSE.

$$MSE = \left(\frac{1}{n} \right) \sum_{i=0}^n (y_i - x_i)^2 \quad (2.16)$$

La Raíz del Error Cuadrático Medio (RMSE – *Root Mean Square Error* en Inglés) al igual que la métrica MSE penaliza los errores altos, solo que para esta métrica se eliminan los cuadrados obteniendo la raíz del promedio obtenido con MSE. La Ecuación 2.17 define el calculo de RMSE.

$$RMSE = \sqrt{\left(\frac{1}{n} \right) \sum_{i=0}^n (y_i - x_i)^2} \quad (2.17)$$

2.4. Aprendizaje profundo

Esta Sección introduce en los temas principales del aprendizaje profundo que sirven como base para comprender como funcionan.

2.4.1. Redes neuronales

Una red neuronal se llama así porque pretende modelar el cerebro en código informático. El objetivo final es crear una “inteligencia general artificial”, en otras palabras, un programa que puede aprender todo lo que una persona podría aprender.

Las redes neuronales artificiales son una clase de algoritmos de aprendizaje máquina que aprenden de los datos y se especializan en el reconocimiento de patrones (Rosebrock, 2017).

Una red neuronal es una función flexible que adapta de manera autónoma su comportamiento para satisfacer la relación entre las entradas y los resultados esperados; por lo que se

lee ha denominado como un aproximador universal (Goyal et al., 2018).

2.4.2. Perceptrón

Rosenblatt publicó el algoritmo seminal Perceptrón: este modelo es capaz de aprender automáticamente los pesos necesarios para clasificar una entrada.

Perceptrón toma n entradas y produce una única salida binaria si la suma es mayor que el valor de activación. Se dice que la neurona se “dispara” siempre que se excede el valor de activación y se comporta como una función escalonada, lo cual se muestra en la Ecuación 2.18 (Goyal et al., 2018).

$$f(x) = \begin{cases} 0 & x - u < 0 \\ 1 & x - u \geq 0 \end{cases} \quad (2.18)$$

Minsky & Papert demostraron que un perceptrón con una función de activación lineal es simplemente un clasificador lineal, incapaz de resolver problemas no lineales (Rosebrock, 2017). El ejemplo de un problema no lineal es el conjunto de datos XOR Figura 2.8. Esta limitante provoca una disminución en el interés por el modelo Perceptrón.

Perceptrón se puede convertir fácilmente en un algoritmo lineal que procesa un flujo de ejemplos, actualizando el vector de peso solo si el último ejemplo recibido está mal clasificado. Se garantiza que el perceptrón convergerá en una solución si los datos de entrenamiento son linealmente separables, pero de lo contrario no convergerán(Flach, 2012).

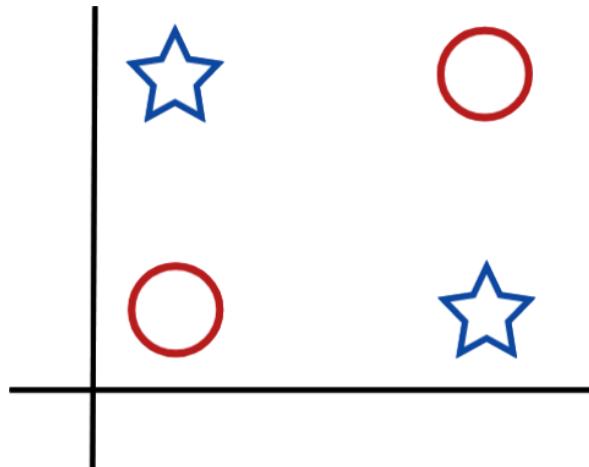


Figura 2.8: Problema XOR, no es linealmente separable.

2.4.3. Redes Neuronales Multicapa

Al encontrar que el modelo Perceptrón solo puede resolver problemas lineales las investigaciones para esta área disminuyeron considerablemente, sin embargo, Rumelhart & McClelland encontraron que este problema puede ser resuelto al unir múltiples perceptrones uno detrás del otro en diferentes combinaciones.

Los perceptrones multicapa o MLP (*Multilayer Perceptron*) son una red neuronal de propagación hacia adelante y se componen de múltiples perceptrones conectados entre sí y que operan en funciones de activación distintivas para permitir mejores mecanismos de aprendizaje. La arquitectura perceptrón multicapa cuenta como mínimo con tres capas: una capa de entrada, una o más capas ocultas y una capa de salida (Swamynathan, 2017) Figura 2.9.

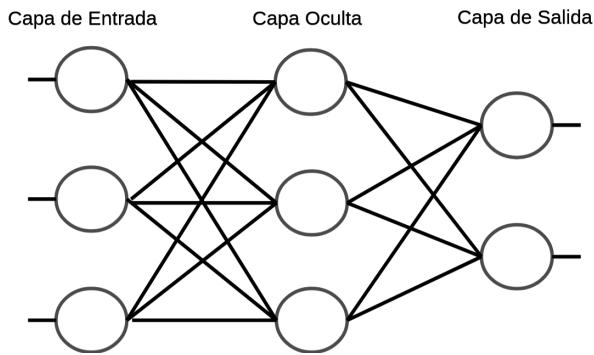


Figura 2.9: Representación básica de red neuronal multicapa.

Los MLP también se conocen como aproximadores universales, ya que pueden encontrar la relación entre los valores de entrada y los objetivos, utilizando una cantidad suficiente de neuronas en la capa oculta, alterando los pesos o utilizando datos de entrenamiento adicionales para aproximar la función dada hasta cualquier nivel de precisión. A menudo, con el grado de libertad dado a un MLP, puede superar a la red MLP básica, introduciendo más capas ocultas, con menos neuronas en cada una de las capas ocultas y pesos óptimos. Esto ayuda en el proceso de generalización del modelo (Goyal et al., 2018).

2.4.4. Propagación hacia atrás (Backpropagation)

En las redes neuronales es complicado ajustar sus pesos de entrenamiento por la gran cantidad de conexiones y sus múltiples niveles. Para lograr esta tarea, se utiliza el algoritmo

de propagación hacia atrás, creado por Rumelhart & McClelland en 1986.

En el algoritmo propagación hacia atrás se calcula qué tan rápido cambia el error a medida que cambia una capa oculta. A partir de ahí, se puede averiguar qué tan rápido cambia el error cuando varía el peso de una conexión individual. Básicamente, pretende encontrar el camino de descenso más pronunciado. Se comienza calculando las derivadas del error con respecto a un solo ejemplo de entrenamiento. Una vez que se tienen las derivadas de error para una capa de unidades ocultas, se utilizan para calcular las derivadas de error para las unidades de la capa siguiente. Y una vez que se encuentran las derivadas del error para las actividades de las unidades ocultas, es bastante fácil obtener las derivadas del error para los pesos que conducen a una unidad oculta (Buduma, 2017).

En resumen, el algoritmo de propagación hacia atrás consiste en dos pasos como muestra la Figura 2.10:

- Propagación hacia adelante: Se calculan los pesos de cada una de las neuronas, desde la capa de entrada hasta la capa de salida.
- Propagación hacia atrás: Se calcula el error y se actualizan los pesos, desde la capa de salida hasta la capa de entrada.

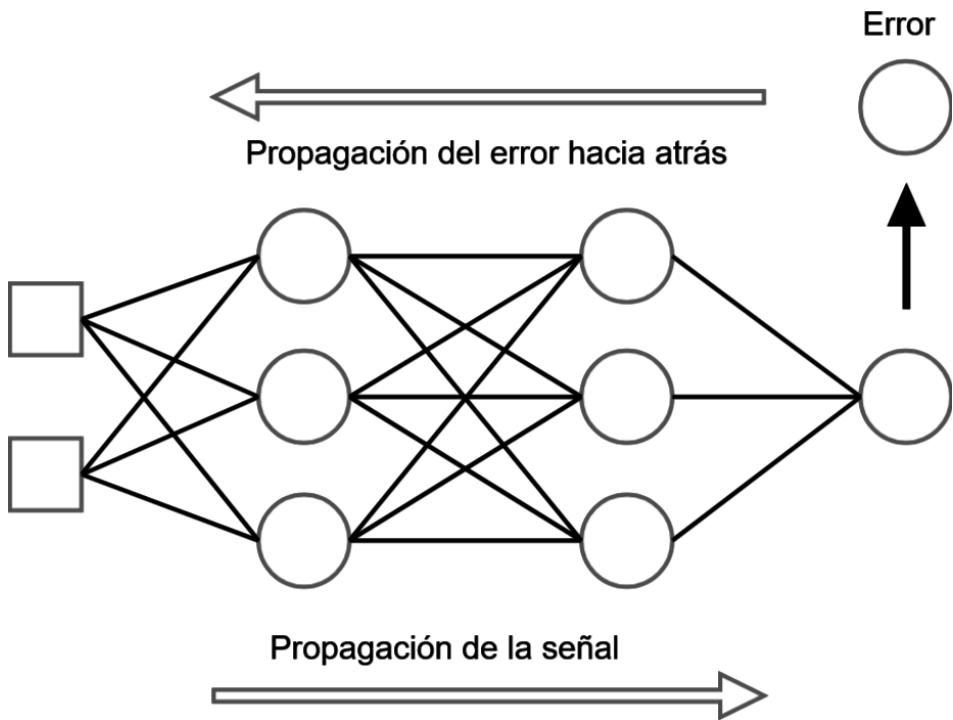


Figura 2.10: Propagación hacia adelante y propagación hacia atrás.

2.4.5. Descenso del Gradiente

El método del descenso del gradiente es un algoritmo iterativo que minimiza una función de pérdida actualizando posteriormente los parámetros de la función.

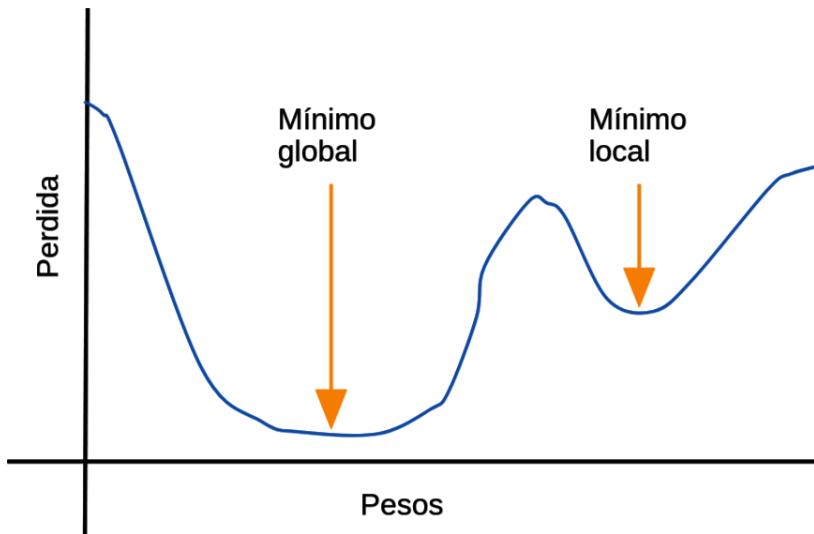


Figura 2.11: Mínimo local y global.

La Figura 2.11 muestra cómo se puede tener múltiples picos y valles (Máximos y mínimos) para los valores de pérdida. El valor ideal que se desean obtener es el mínimo global, asegurando que los parámetros tomen los valores óptimos posibles.

El problema es que no se cuenta con una vista general en la que sea posible encontrar de manera sencilla el mínimo global ya que en realidad se inicializa en un lugar aleatorio sin saber dónde está el mínimo global y se avanza hacia una pérdida mínima sin subir accidentalmente a la cima de un máximo local.

2.4.6. Tasa de aprendizaje

La tasa de aprendizaje es uno de los hiperparámetros más importantes. Tiene un fuerte impacto tanto en la estabilidad como en la eficiencia de los tiempos de entrenamiento y no hay una forma fija de encontrar la más adecuada. La tasa de aprendizaje es la magnitud del ajuste de pesos durante el entrenamiento de la red con el fin de minimizar el error (Partida, 2020).

La Figura 2.12 muestra que, si la tasa es demasiado grande, el entrenamiento es inestable. Si la tasa de aprendizaje es demasiado pequeña, el entrenamiento puede tomar más tiempo para llegar al ideal.

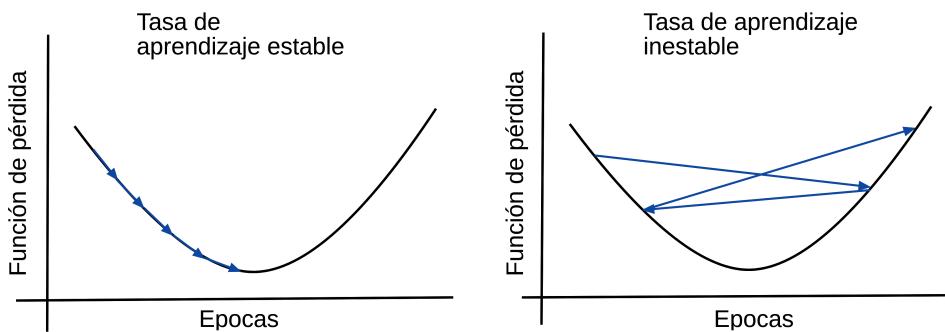


Figura 2.12: Tasa de aprendizaje.

Preferiblemente, la pérdida de entrenamiento y validación casi se imitan entre sí, con solo una pequeña brecha entre la pérdida de entrenamiento y la pérdida de validación, lo que indica un pequeño sobreajuste. Siempre que la brecha no aumente drásticamente, existe un nivel aceptable de sobreajuste. Por otro lado, si no logra mantener esta brecha y las pérdidas de entrenamiento y validación se separan drásticamente, entonces se sabe que se corre el

riesgo de sobreajuste. Una vez que la pérdida de validación comienza a aumentar, se sabe que estamos muy sobreajustados (Rosebrock, 2017).

2.4.7. Funciones de activación

Las funciones de activación son una función de escalar a escalar, que produce la activación de la neurona. Las funciones de activación son usadas para propagar la salida de los nodos de una capa hacia la siguiente capa y para que las neuronas ocultas en una red neuronal para introducir la no linealidad del modelado de la red (Patterson & Gibson, 2017).

Las funciones de activación definen el rango de valores que dará como resultado una de las capas de nuestra red neuronal, a partir de una función matemática.

2.4.7.1. Función lineal

La función lineal regresa como salida valores en el rango de $[-\infty, \infty]$ y está dada por la Ecuación 2.19:

$$f(x) = Wx \quad (2.19)$$

La Figura 2.13 muestra la gráfica generada por la Ecuación 2.19.



Figura 2.13: Gráfica de función lineal.

2.4.7.2. Sigmoidal

La función sigmoidal tiene como salida valores en el rango de $[0, 1]$ y está definida por la Ecuación 2.20:

$$s(x) = \frac{1}{1 + e^{-x}} \quad (2.20)$$

La Figura 2.14 muestra la gráfica generada por la Ecuación 2.20.

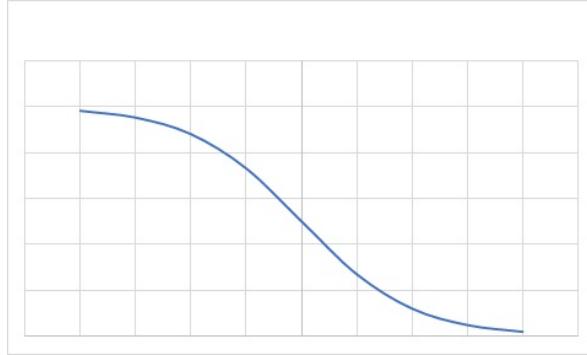


Figura 2.14: Gráfica de función Sigmoidal.

2.4.7.3. Tangente hiperbólica

La función tangente hiperbólica proporciona como salida el rango $[-1, 1]$ y está definida por la Ecuación 2.21.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.21)$$

La Figura 2.15 muestra la gráfica generada por la Ecuación 2.21.

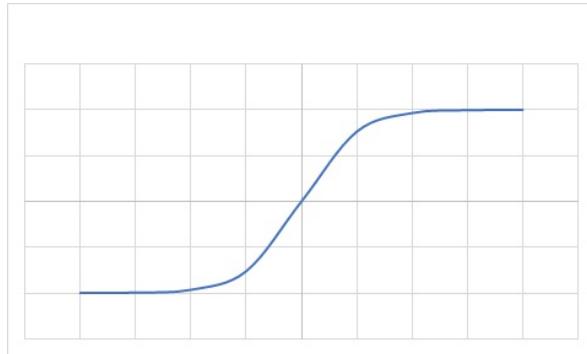


Figura 2.15: Gráfica de función hiperbólica.

2.4.7.4. Softmax

La función de activación softmax devuelve la distribución de probabilidad sobre clases de salida mutuamente excluyentes. La cual está definida por la siguiente Ecuación 2.22:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2.22)$$

donde \vec{z} es el vector de entrada.

z_i son los valores del vector de entrada, e^{z_i} es la función exponencial estándar que se aplica a cada elemento del vector de entrada. $\sum_{j=1}^K e^{z_j}$ es el término de normalización, asegura que todos los valores de salida de la función sumen 1 y cada uno esté en el rango $(0, 1)$, constituyendo así una distribución de probabilidad válida. Para finalmente, k es el número de clases del clasificador multi-clases.

2.4.7.5. ReLU

La función Rectificador Lineal(ReLU) regresa como resultado valores en el rango de los números reales positivos $[0, \infty]$, como muestra la Ecuación 2.23:

$$relu(x) = \max(0, x) \quad (2.23)$$

La Figura 2.16 muestra la gráfica generada por la Ecuación 2.23.

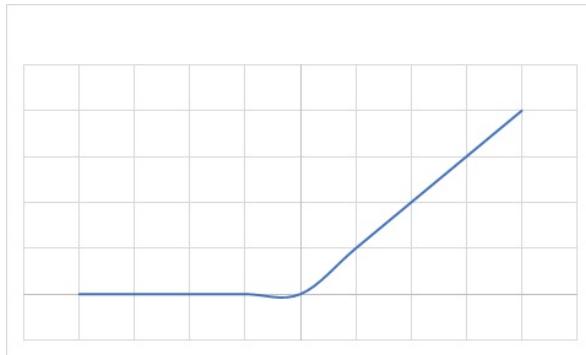


Figura 2.16: Gráfica de función ReLU.

2.4.8. Funciones de pérdida

Una función de pérdida indica la pérdida de predicción de \hat{y} cuando la salida real es y . El objetivo del entrenamiento es entonces minimizar la pérdida en los diferentes ejemplos de entrenamiento. La función asigna una puntuación numérica a la salida de la red \hat{y} dada la verdadera salida esperada y (Goldberg, 2017).

Dependiendo del problema es el tipo de función de pérdida que se utiliza. Para un problema de regresión se busca tener la pérdida lo más cercana a cero y para un problema de clasificación se busca tener el valor más alto o muy cercano a 1 para cada una de las clases para la cual fue entrenada la red neuronal, siempre cuidando que la red neuronal no tenga sobreajuste.

Algunas de las funciones de pérdida para problemas de regresión es el Error Cuadrático Medio (MSE, *Mean squared error loss*) definido por la Ecuación 2.24 y el Error Absoluto Medio (MAE, *Mean absolute error loss*) definido por la Ecuación 2.25.

$$MSE = \left(\frac{1}{n} \right) \sum_{i=0}^n (y_i - x_i)^2 \quad (2.24)$$

$$MAE = \left(\frac{1}{n} \right) \sum_{i=0}^n |y_i - x_i| \quad (2.25)$$

Mientras que para problemas de clasificación la pérdida de bisagra (*Hinge loss*) es la función de pérdida mas común y definida por la Ecuación 2.26.

$$Hinge Loss = \max(0, 1-t) \quad (2.26)$$

2.4.9. Optimizadores

Para el entrenamiento de una red neuronal es necesario adaptar los hiperparámetros de entrada para generar un buen modelo, en ocasiones suele ser una tarea complicada ya que no hay un manual que indique como configurar los hiperparámetros de tal manera que el modelo proporcione los mejores resultados, por lo tanto, es una tarea de prueba y error.

La tasa de aprendizaje suele ser uno de los hiperparámetros más relevantes afectando

directamente en el tiempo de entrenamiento de la red. Con ayuda del descenso del gradiente, este hiperparámetro puede resultar de mucha ayuda para el entrenamiento, sin embargo, tiene la limitante de que la tasa de aprendizaje es un valor fijo, por otra parte, existen algoritmos que ayudan a reducir el valor de la tasa de aprendizaje para mejorar los resultados del modelo.

Los optimizadores adaptativos son mejoras al algoritmo del descenso de gradiente los cuales cambian el valor de la tasa de aprendizaje para cada parámetro, tomando como referencia los resultados obtenidos. Algunos de los optimizadores más populares son los siguientes:

- AdaGrad (*Adaptive Gradient Algorithm*): Este algoritmo se encarga de modificar el valor de la taza de aprendizaje para mejorar los resultados y el tiempo de entrenamiento.
- AdaDelta: Es una extensión de AdaGrad para reducir el cambio drástico de la tasa de aprendizaje.
- RMSProp (*Root Mean Squared Propagation*): Al igual que AdaDelta, RMSProp ayuda a reducir el cambio drástico de la tasa de aprendizaje del AdaGrad.

2.5. Tecnologías utilizadas

Para el desarrollo de este trabajo se utilizaron múltiples herramientas como lo son Git, SublimeText 3, Sublime Merge y los repositorios de Github todo bajo el sistema operativo Ubuntu 20.04.

A continuación se presentan las herramientas de software de mayor impacto para el desarrollo de este trabajo.

2.5.1. Python

Python (van Rossum, 1994) es un lenguaje de programación de alto nivel, interpretado y multipropósito. Python es muy utilizado por la comunidad al ser un leguaje muy fácil de aprender y utilizar por ser un lenguaje de tipado dinámico, además de ser de código abierto.

Python ha tomado popularidad para el desarrollo en *deep learning*, por su facilidad de uso y la gran variedad de bibliotecas con las que cuenta como TensorFlow, PyTorch, Numpy, Scikit, entre otros.

2.5.2. Contenedores Docker

Un contenedor es una unidad de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y confiable de un entorno informático a otro. Una imagen de contenedor de Docker es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, tiempo de ejecución, herramientas del sistema, bibliotecas del sistema y configuraciones (*What is a container?* s.f.).

La Figura 2.17 muestra las capas desde la infraestructura hasta los contenedores ejecutándose.

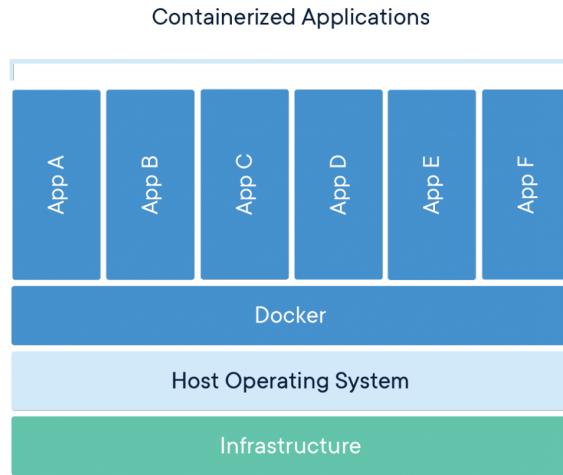


Figura 2.17: Representación de un contenedor (*What is a container?* s.f.).

Docker fue la principal herramienta utilizada para el desarrollo de este trabajo, puesto que, sirvió para replicar múltiples trabajos del estado del arte sin necesidad de complicadas configuraciones al equipo utilizado, además de transferir los experimentos a otro equipo al solo utilizar la imagen creada para este experimento.

2.5.3. 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.

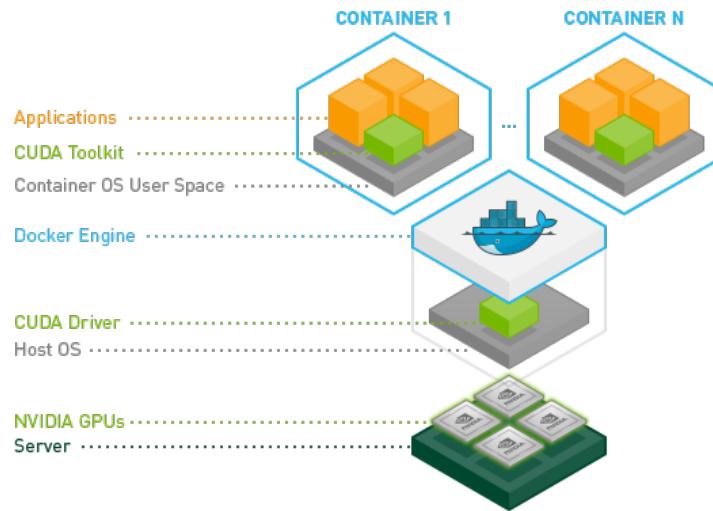


Figura 2.18: Arquitectura de Nvidia Docker (*Overview s.f.*).

Capítulo 3

Estado del Arte

Actualmente existen múltiples trabajos que determinan la velocidad de objetos, cada uno de una manera muy particular. En este capítulo se muestran algunos trabajos divididos por la metodología utilizada, los cuales se tomaron como referencia para desarrollar el presente trabajo.

3.1. Detección de acuerdo a simple cálculo con fórmula física

Singh (Singh, 2015) propone determinar la velocidad, aceleración y ángulo de un objeto en una secuencia de imágenes. Para esto se deben cumplir con las siguientes características:

- Se conoce un rango aproximado de valores RGB del objeto.
- Se conoce la velocidad a la que la cámara está tomando imágenes.
- Se conoce aproximadamente el tamaño de la imagen.
- La imagen es de color uniforme.
- El fondo es de color uniforme y no es del color del objeto.

Para determinar los valores buscados se dibuja un punto en el centroide del objeto en la primera imagen y en la segunda imagen, a partir de tener los dos centroides es posible calcular la velocidad, aceleración y ángulo del objeto con las Ecuaciones 3.1, 3.2 y 3.3:

$$\frac{\sqrt{(x2 - x1)^2 + (y2 - y1)^2}}{FPS} \quad (3.1)$$

$$\frac{v2 - v1}{FPS} \quad (3.2)$$

$$\tan^{-1} \frac{y2 - y1}{x2 - x1} \quad (3.3)$$

3.2. Detección de acuerdo a región de interés

La región de interés es una metodología la cual se utiliza una máscara para solo tomar en cuenta el área con la cual se va a trabajar. S. Li et al. (S. Li et al., 2014) desarrollan un sistema para recopilar datos de tráfico a partir de video, el sistema reconoce y realiza el seguimiento de múltiples vehículos y sus velocidades medias. Proponen la sustracción de fondo adaptativo basado en imágenes en color, se eliminan sombras y la configuración de la región de detección para mejorar la robustez del sistema. El conteo de vehículos tiene una precisión de 97.4 %, un error de clasificación de 8.3 % y un error absoluto de 2.3 kilómetros por hora.

Kurniawan et al. (Kurniawan et al., 2018) proponen un sistema que puede monitorear la velocidad de algunos vehículos y determinar qué vehículos han superado la velocidad máxima permitida mediante el seguimiento del vehículo utilizando un método basado en coincidencias. Se utilizó el método de transformación proyectiva para calcular la velocidad de un vehículo. Este método se utiliza para transformar la imagen capturada en la imagen de la vista superior en forma de rectángulo. Según los resultados de las pruebas en el sistema a una velocidad de 30 fotogramas por segundo, la precisión de la velocidad de cálculo es de 97.01 % cuando no hay sombra y 83.86 % con sombra. Este sistema también puede determinar el tipo de vehículo de automóvil o motocicleta detectado que se rompe con una precisión del 89.62 %.

Jalalat et al. (Jalalat et al., 2016) proponen la detección de vehículos y su velocidad con un clasificador en cascada, basado en las características de Haar. Un algoritmo de segmentación de primer plano para distinguir los vehículos en movimiento del fondo y limpiar los resultados de detección para combinarlos con resultados de seguimiento basados en el filtro Kalman y

algoritmo de asignación de Munkres. Para lograr una medición de velocidad, se aplica una técnica eficiente de coincidencia de sub píxeles junto con datos estéreo para calcular los desplazamientos de vehículos por fotograma.

3.3. Detección usando cámaras estéreo y placa del vehículo

En algunos trabajos utilizan hardware especializado como las cámaras stereo las cuales con ayuda de dos cámaras posicionadas a una distancia específica son capaces de determinar la profundidad de la escena, algunos trabajos utilizan estas cámaras y características comunes de un vehículo como la placa para calcular la velocidad a la que viajan.

Yang, M. Li et al. (Yang, M. Li et al., 2019) proponen el uso de cámaras estéreo calibradas con las cuales estiman la velocidad de vehículos. Este desarrollo usa un sistema optimizado de detectores de múltiples áreas de un solo disparo que puede detectar de manera eficiente las matrículas en las dos vistas capturadas videos estéreo. El sistema funciona solo en el área de la placa, se calculan las coordenadas en un plano 3D correspondientes a los puntos de coincidencia estéreo. La velocidad se mide dividiendo la distancia entre dos puntos 3D por intervalo de cuadros. El sistema tiene un error de velocidad de -1.6 a +1.1 Km y una tasa de error máxima del 3.80 %.

Luvizon et al. (Luvizon et al., 2014) nos muestran un sistema que utiliza la detección de texto para localizar las matrículas de los vehículos, la cual es utilizada para seleccionar las características más estables para el seguimiento. La velocidad del vehículo se estima comparando la trayectoria de las características rastreadas con medidas conocidas del mundo real. En los experimentos realizados las velocidades de los vehículos se estimaron con un error absoluto medio de 0.59 km/h.

Fernández-Llorca et al. (Fernández-Llorca et al., 2016) determinan la velocidad de vehículos con el uso de dos cámaras montadas en un poste fijo. Usando diferentes distancias focales y orientaciones, cada cámara apunta a un tramo diferente de la carretera de unos pocos metros, lo que implica un escenario desafiante con la estimación de la distancia los errores deben ser del orden de centímetros. El sistema propuesto determina la distancia relativa a los

vehículos con respecto a las cámaras utilizando la placa como referencia se demuestra que existe una geometría específica entre las cámaras que minimiza el error de velocidad. Los resultados obtenidos validan la propuesta con errores de velocidad máxima de menos 3 k/h a velocidades de hasta 80 k/h.

Yang, Luo et al. (Yang, Luo et al., 2021), Yang, Q. Li et al. (Yang, Q. Li et al., 2021), determinan la velocidad con cámaras estéreo que están ubicadas de fijas en un lugar estratégico de la ciudad. Las cámaras estéreo son las encargadas de identificar la matrícula, los faros y el logo, ya que estas se toman como las características principales, ya que siempre aparecen en un vehículo; estas características se utilizan como referencia para determinar la velocidad. En Vakili et al. (Vakili et al., 2020), se usa una estrategia similar identificando solo la placa como la característica principal y usándola para determinar la velocidad del vehículo. Sin embargo, Vakili et al., usa solamente una cámara, en lugar de cámaras estéreo.

3.4. Detección de acuerdo a líneas carreteras

Las líneas carreteras sirven como punto de referencia para determinar la velocidad, ya que estas tienen una distancia fija, la cual puede ser utilizada como referencia para determinar la distancia que recorrió el vehículo al que se desea calcular la velocidad.

Schoepflin & Dailey (Schoepflin & Dailey, 2003) muestran un algoritmo de tres etapas para calibrar cámaras de tráfico en carretera y rastrear vehículos para crear un sensor de velocidad del tráfico. El algoritmo primero estima la posición de la cámara en relación con la carretera usando el movimiento y los bordes de los vehículos. Dada la posición de la cámara, el algoritmo calibra la cámara estimando los límites del carril y el punto de fuga de las líneas a lo largo de la carretera. El algoritmo transforma las coordenadas de la imagen del rastreador de vehículos en coordenadas del mundo real utilizando un modelo de cámara simplificado.

Bevilacqua et al. (Bevilacqua et al., 2016) implementan la transformada de Hough para detectar las líneas carreteras para ser utilizadas como referencia. Para el seguimiento de los vehículos utilizan el popular algoritmo de Kanade-Lucas-Tomasi, para calcular la velocidad se determina la longitud de la región de interés en pixeles la cual equivale a la longitud en metros para después obtener la velocidad con respecto al tiempo en segundos.

Anil Rao et al. (Anil Rao et al., 2015) proponen un framework híbrido para el seguimiento de múltiples vehículos con la combinación del filtro Kalman y el algoritmo húngaro para resolver el problema de las obstrucciones. La estimación de la velocidad puede ser realizada sin la necesidad de calibrar las cámaras a utilizar, ya que utiliza las líneas del pavimento para realizar una estimación con un error máximo de 3 kilómetros por hora.

3.5. Detección de acuerdo a centroide del vehículo

Cuando no se tiene la distancia real que recorre un vehículo es común utilizar como referencia el centroide del área detectada de un vehículo ya que tomando dos imágenes como referencia esta establece una distancia recorrida en pixeles.

Kamoji et al. (Kamoji et al., 2020) proponen procesar un video cuadro a cuadro, en el cual cada cuadro se procesa con *Image Enhancement* para mejorar las características de la imagen, el vehículo se identifica con un cuadro delimitador aplicado al vehículo, para notar el movimiento en cada cuadro, el movimiento del vehículo se anota con cambio en el píxel y el cálculo de la velocidad se determina usando la fórmula de distancia considerando pixeles por metro.

Lee (Lee, 2021) desarrollo un módulo para medir la distancia de vehículos en varios carriles simultáneamente usando un dron. El dron está montado con dos sensores LiDAR, y cada sensor emite un punto frontal y un punto trasero, donde los vehículos son detectados. La velocidad del vehículo se estima utilizando la distancia entre el punto delantero y trasero por el cual pasa el vehículo y el tiempo que tarda en pasar por ambos puntos.

3.6. Detección usando redes neuronales

Las redes neuronales son ampliamente utilizadas ya que el poder de cómputo ha aumentado considerablemente, para determinar la velocidad las redes neuronales son utilizadas tanto directa como indirectamente algunos trabajos las utilizan para determinar la velocidad a partir de características obtenidas con hardware especializado, mientras que otros solo la utilizan en ciertas partes como la detección de objetos.

YOLO (Redmon et al., 2016) implementa la detección de objetos como un problema de regresión a cuadros delimitadores separados espacialmente y probabilidades de clase asociadas. Una sola red neuronal predice el área y las probabilidades de múltiples clases a partir de una imagen. Esto significa que YOLO razona globalmente sobre la imagen completa y todos los objetos de la imagen. Dado que toda la detección es una sola red, afecta directamente en el rendimiento de la detección. YOLO es implementada como una red neuronal convolucional y es evaluada usando el conjunto de datos PASCAL VOC. Las capas convolucionales iniciales de la red extraen características de la imagen, mientras que las capas completamente conectadas predicen las probabilidades y coordenadas de salida. Según los autores la red YOLO es extremadamente rápida lo cual la hace ideal para detecciones en tiempo real, procesando imágenes a 45 fotogramas por segundo. Una implementación reducida de la red puede procesar 155 fotogramas por segundo. Por otra parte, YOLO comete más errores de localización, al precio de disminuir la predicción de falsos positivos, superando otros métodos de predicción como DPM y RCNN.

Moritz Kampelmuhler & Feichtenhofer (Moritz Kampelmuhler & Feichtenhofer, 2018) determinan la velocidad relativa de un vehículo utilizando una red neuronal perceptrón multicapa la cual fue entrenada usando secuencias de imágenes, obteniendo buenos resultados con un error promedio de 1.12 m/s, el cual es comparable con un radar LiDAR con un error de 0.71 m/s. Para determinar la velocidad se identificaron 3 características esenciales: la trayectoria del vehículo en un plano 2D, la profundidad y estimaciones de flujo óptico entre imágenes consecutivas. Al identificar un vehículo en una secuencia de imágenes podemos identificar que este, aumenta o disminuye su tamaño dependiendo la distancia que tiene con respecto al observador, esto complica el aprendizaje para la red neuronal por lo cual se opta por separar la información en cerca, medio y lejos, y entrenar un modelo para cada distancia.

Yaqi Zhang & Liu (Yaqi Zhang & Liu, 2017) realizaron experimentos para obtener la aceleración, la velocidad y la velocidad angular de un vehículo en movimiento usando cámaras de video posicionadas frente al área de interés. Para sus experimentos utilizaron el conjunto de datos KITTI el cual cuenta con imágenes estereoscópicas de la ciudad, así como arquitecturas CNN y la derivación y combinación de canales de entrada para identificar la dirección del flujo y las máscaras de objetos. Se creó una arquitectura CNN para cada característica

buscada, aceleración, velocidad y velocidad angular. Se tomó como línea base una arquitectura CNN de dos capas, que consta de dos capas conv-relu-batchnorm-maxpooling y 2 capas afines-relu-batchnorm-dropout. Este proyecto compara los resultados para cuatro arquitecturas CNN: línea de base, AlexNet, ResNet y AlexNet con el aprendizaje por transferencia. Los resultados muestran que para el conjunto de entrenamiento ResNet tiene los mejores resultados seguidos por AlexNet con el aprendizaje por transferencia, AlexNet y la línea base. Sin embargo, para el conjunto de validación los mejores resultados los muestra la línea base, puesto que el entrenamiento de este modelo es más rápido y es posible buscar los hiperparámetros para los mejores resultados. AlexNet con el aprendizaje por transferencia y AlexNet son los segundos mejores y ResNet tiene los peores resultados para el conjunto de validación.

Dong et al. (Dong et al., 2019) proponen un método alternativo de extremo a extremo basado en redes convolucionales tridimensionales. El método propuesto basa la estimación de la velocidad promedio del vehículo en información de imágenes de video. El método se caracteriza por las siguientes tres características. Primero, el uso de bloques no locales en el modelo para capturar mejor la dependencia espacio-temporal de largo alcance. En segundo lugar, el uso de flujo óptico como entrada en el modelo. El flujo óptico incluye la información sobre la velocidad y la dirección del movimiento de los píxeles en una imagen. En tercer lugar, se construyó una red convolucional de múltiples escalas. Esta red extrae información sobre diversas características de los vehículos en movimiento. El método propuesto muestra resultados experimentales prometedores en un conjunto de datos de uso común con un error absoluto medio (MAE) de 2.71 km/h y un error cuadrático medio (MSE) de 14.62.

Song et al. (Song et al., 2020) determinan la velocidad y distancia de un vehículo usando dos fotogramas consecutivos, de los cuales obtienen las características profundas, la geometría de la escena y el flujo óptico temporal. La arquitectura de esta red neuronal profunda se basó en la arquitectura de PWC-Net, la cual es una red de estimación de flujo óptico eficiente. Se extraen diferentes características de diferentes capas de la red para estimar la distancia y la velocidad del vehículo. En primer lugar, se presenta el modelo de regresión de distancia utilizando características profundas y geométricas espaciales de vehículos. Después de eso, se propone el método de estimación de la velocidad con características geométricas adicionales y una pista de flujo óptico temporal. Por último, detallan la canalización de la red

centrada en vehículos para reducir la perspectiva y la influencia del movimiento.

Bell et al. (Bell et al., 2020) utilizan una cámara de video para estimar la velocidad vehicular. Con el uso de YOLOv2 se encargan de detectar los vehículos de la escena, mientras que el algoritmo de seguimiento en tiempo real en línea simple para el seguimiento de vehículos. La transformación del metraje de la cámara al Sistema de coordenadas de cuadrícula nacional británico, permitió la derivación de distancias del mundo real en la superficie plana de la carretera y las estimaciones simultáneas de la velocidad del vehículo. Las estimaciones lograron una raíz del error cuadrático medio y un error porcentual absoluto medio de 0.625 m/s y 20.922 %, respectivamente.

Loor (Loor, 2017) busca determinar la velocidad de los objetos utilizando dos imágenes consecutivas. Para encontrar la velocidad el autor tomo como base la red neuronal FlowNet la cual ha aprendido a encontrar el flujo óptico en dos imágenes. FlowNet tiene dos arquitecturas de red diferentes: una arquitectura simple y una arquitectura de correlación. Ambas arquitecturas se basan en la red totalmente convolucional, lo que significa que esta red puede utilizar cualquier tamaño de imagen de entrada para determinar su flujo óptico. Una FlowNetSimple se entrena usando dos imágenes que se apilan juntas, dando lugar a una imagen de seis dimensiones. Las siguientes capas se han diseñado como una red genérica, para permitir que la red decida cómo aprender el flujo óptico. En la FlowNetCorr se combinan las dos imágenes en una etapa posterior, después de que el modelo haya creado representaciones significativas de cada imagen por separado. Para ayudar al modelo a aprender las correspondencias entre la representación de características de las imágenes, se introdujo una capa de correlación. Para este trabajo se crearon dos modelos basados en FlowNet, TimeNet y SpeedNet. TimeNet es un clasificador que puede predecir los tiempos entre fotogramas en milisegundos. SpeedNet es un modelo de regresión que genera una predicción de velocidad en forma de un solo valor. Tanto el modelo TimeNet como SpeedNet demostraron dar resultados aceptables cuando se tiene una velocidad baja, empeorando cuando la velocidad aumenta.

Peng et al. (Peng et al., 2019) determinan la velocidad de vehículos en movimiento con el uso de cámaras estereos montados en drones. Su método integra Mask-R-CNN y K-Means con el algoritmo de pirámide Lucas Kanade. Mask-R-CNN se utiliza para reconocer los objetos que tienen movimientos en relación con el suelo y los cubre con máscaras para mejorar la

similitud entre píxeles y reducir los impactos de los ruidosos píxeles en movimiento. Luego, se utiliza el algoritmo piramidal de Lucas Kanade para calcular el valor de flujo óptico. Finalmente, el valor es agrupado por el algoritmo K-Means para abandonar los valores atípicos, y la velocidad del vehículo es calculada por el flujo óptico procesado. En sus experimentos demuestran que el uso combinado de Lucas Kanade , Mask-R-CNN y K-Means, mejoran sus resultados en comparación del uso de estos métodos por separado o el uso combinado de solo dos de ellos, aun habiendo realizado experimentos en circunstancias normales, con muchos objetos en movimiento y en circunstancias de poca luz.

3.7. Resumen

En este capítulo se presentaron los diferentes trabajos relacionados al análisis vehicular enfocándose en el cálculo de la velocidad, así como los trabajos de mayor impacto para realizar esta tarea o parte de ella utilizando cámaras de video.

La Tabla 3.1 muestra la agrupación de los trabajos escritos en este capítulo, comenzando por Singh quien utiliza la metodología mas simple, la cual sirve como punto de partida utilizando las fórmulas físicas. Fernández-Llorca et al., Yang, M. Li et al., Yang, Luo et al., Yang, Q. Li et al., Luvizon et al. y Vakili et al. hacen uso de una cámara, dos cámaras o una cámara estéreo para tomar videos donde pasan los vehículos a los que se les busca determinar su velocidad, estos trabajos utilizan como referencia la placa del vehículo o características que podemos encontrar frecuentemente en un vehículo como las luces, gracias a estas características es posible determinar la trayectoria y las estimaciones de la distancia recorrida.

Bevilacqua et al., Schoepflin & Dailey y Anil Rao et al. utilizan una cámara de video con la cual identifican las líneas carreteras, las cuales son utilizadas como punto de referencia para estimar la distancia recorrida por los vehículos y así calcular la velocidad de los mismos.

Kamoji et al. utiliza una cámara y el centroide del vehículo como punto de referencia, por otra parte Lee también toma como referencia el centroide del vehículo, pero utilizan un dron con dos sistemas LiDAR con los cuales detectan los vehículos cuando pasan por dos puntos para calcular la velocidad a la que pasan.

S. Li et al., Kurniawan et al. y Jalalat et al. hacen uso de una cámara de video para procesar

las imágenes enfocándose solo en el área por la que pasan los vehículos y con un proceso mas completo eliminar completamente todo el fondo dejando solo los vehículos.

Bell et al., Dong et al., Burnett et al., Moritz Kampelmuhler & Feichtenhofer, Song et al., Yaqi Zhang & Liu y Loor utilizan métodos de inteligencia artificial desde las mas simples como los son las perceptrón multi capa y las redes neuronales convoluciones hasta redes como Faster R-CNN, SqueezeDet, PWC-Net, AlexNet y FlowNet.

Por último el trabajo de Redmon et al. presentan YOLO, la red neuronal YOLO es capaz de identificar múltiples objetos en una imagen con una sola inferencia. Este trabajo no esta enfocado al análisis de trafico vehicular, sin embargo, es ampliamente utilizado en trabajos relacionados por lo cual es importante mencionarlo.

Tabla 3.1: Resumen estado del arte

	Hardware	Referencia para estimación	Red Neuronal
Singh		Los objetos y el fondo deben ser de un color en concreto. Centroide	
Fernández-Llorca et al.	Dos cámaras	Placa vehicular	
Yang, M. Li et al.	Cámaras estéreo	Placa vehicular	
Yang, Luo et al.	Cámaras estéreo	Placa vehicular	
Yang, Q. Li et al.	Cámaras estéreo	Placa vehicular	
Luvizon et al.	Cámara	Placa vehicular	
Vakili et al.	Cámara	Placa vehicular	
Bevilacqua et al.	Cámara	Lineas carreteras	
Schoepflin & Dailey	Cámara	Lineas carreteras	
Anil Rao et al.	Cámara	Lineas carreteras	
Kamoji et al.	Cámara	Centroide del vehículo	
Lee	Dron LiDAR	Centroide del vehículo	
S. Li et al.	Cámara	Región de interés	
Kurniawan et al.	Cámara	Región de interés	
Jalalat et al.	Cámara	Región de interés	
Bell et al.	Cámara GNSS IMU		YOLOv2 Faster R-CNN
Dong et al.	Cámara		3D CNN
Burnett et al.	GPS LiDAR IMU Cámara		SqueezeDet
Moritz Kampelmuhler & Feichtenhofer	Cámara		MLP
Song et al.	Cámara		PWC-Net
Yaqi Zhang & Liu	Cámara		Faster-RCNN CNN AlexNet
Loor	Cámara		FlowNet TimeNet SpeedNet CNN
Redmon et al.	Imágenes		CNN

Capítulo 4

Metodología

En este capítulo se describe el proceso de obtención de la velocidad de vehículos a partir de secuencias de imágenes. Para realizar esta tarea se recopiló un conjunto de datos (*dataset*) suficiente para realizar la tarea de estimación de la velocidad, a partir del *dataset*. El problema de obtención de la velocidad fue dividido en tres procesos principales (extracción de características, modelo predictivo y obtención de la velocidad). Para la generación del *dataset* fue necesario ir físicamente a una vía terrestre. Una vez en la vialidad, se utilizó una cámara para capturar la secuencias de imágenes y un radar para medir la velocidad del vehículo. El proceso de extracción de características parte de los videos (secuencias de imágenes) para después aplicar técnicas de aprendizaje profundo. Resultando en un archivo con extensión csv con las características esenciales con los vehículos en cuestión. A partir del archivo con extensión csv se aplicaron técnicas estadísticas de correlación con la finalidad de relacionar la distancia en píxeles con una distancia en metros. Para posteriormente estimar la velocidad del vehículo (obtención de la velocidad).

La metodología completa se divide en dos grandes procesos como lo muestra la Figura 4.1. El proceso de obtención del conjunto de datos, que consistió en la generación de muestras de campo con las cuales se realizó esta investigación. Y a partir de las muestras recolectadas se realizó el proceso de obtención de la velocidad.

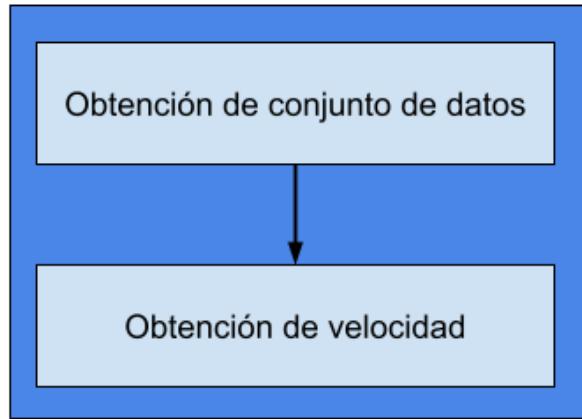


Figura 4.1: Proceso de obtención de la velocidad.

4.1. Obtención de conjunto de datos

La obtención del conjunto de datos se divide en dos etapas, la primera etapa corresponde a la toma de muestras, la cual implica ir al lugar con flujo constante de vehículos para obtener la mayor cantidad de datos. La segunda etapa es la limpieza de las muestras con la cual se busca solo tomar en cuenta los vehículos a los que se logró tomar correctamente la velocidad. La Figura 4.2 muestra las dos etapas internas con las que cuenta la obtención de conjunto de datos.

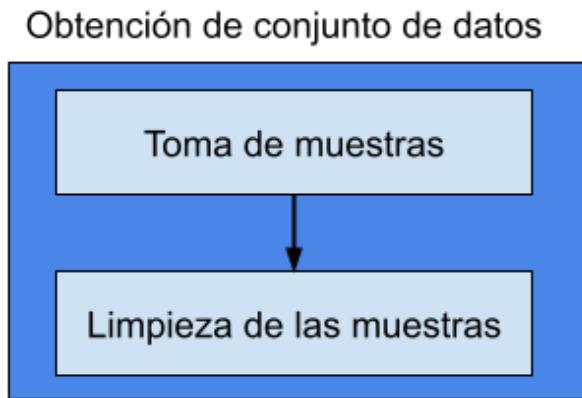


Figura 4.2: Proceso de obtención de conjunto de datos.

4.1.1. Toma de muestras

Para la toma de muestras se requirió de dos dispositivos con los cuales se busca la extracción del conjunto de datos que posteriormente se utilizará como entrenamiento de los modelos predictivos. Uno de los dispositivos es el radar Bushnell (Figura 4.3) el cual tiene una precisión de +/- 1.6 kilómetros por hora.



Figura 4.3: Radar de velocidad Bushnell(*Buy velocity speed gun and more s.f.*).

Mientras que el otro dispositivo es una cámara de video, la cual puede ser un dispositivo especializado para esta tarea o cualquier otro dispositivo capaz de grabar video. Las configuraciones mínimas pueden ser una resolución de 854 x 480 a 30 fotogramas por segundo. Sin embargo, hay que tomar en cuenta que al tener una resolución baja se puede perder calidad en la imagen y tomar un área menor a la deseada. Mientras que tener fotogramas tan bajos ocasionará perder vehículos que pasan a una velocidad más alta. Por otro parte al aumentar la resolución y los fotogramas por segundo ocasiona que al sistema le tome más tiempo procesar el video. Para este caso se utilizó la cámara de Smartphones un Xiaomi Redmi Note 7 y un iPhone X configurados a 60 FPS y una resolución de Full HD (1920 x 1080 pixeles).

Existe un cuidado especial a la hora de posicionar la cámara, ya que no se quiere que los experimentos sean considerados como cálculos de velocidad en 2D, para esto la toma de muestras se realizó en un lugar donde el tráfico vehicular pase con cierto grado de inclinación,

sin apuntar la cámara directamente al costado de donde pasan los vehículos, como muestra la Figura 4.4.



Figura 4.4: Lugar donde se tomaron las muestras.

Cabe mencionar que por practicidad, el proceso de toma de muestras fue realizado, en un mismo lugar de la ciudad de Culiacán, Sinaloa.

Dado que la cámara y el radar de velocidad son dispositivos que no están sincronizados entre sí, fue necesario la implementación de un mecanismo para asociar la lectura del radar con la toma del video.

4.1.2. Limpieza de las muestras

Una vez tomadas las muestras, fue necesario realizar una limpieza de los datos, estos son, los vehículos a los que se les detectó la velocidad utilizando el radar. Con la limpieza de los datos se busca eliminar los vehículos a los que no se les tomó la velocidad y dejando solo aquellos que si fueron considerados.

La toma de las muestras se realizó a partir de dos dispositivos que no están especializados para esta tarea (Cámara de video y radar de velocidad), la limpieza de las muestras ayuda a combinar la información de ambos dispositivos realizando una inspección visual de los videos, en la cual se identifica el segundo del video en el que pasa el vehículo de interés, la velocidad detectada por el radar, el carril por el cual viaja el vehículo y una descripción del

vehículo para futuras referencias. Estos cuatro datos son guardados en un archivo de texto separado por comas (csv) el cual servirá de entrada para el siguiente paso en la metodología.

Determinar la velocidad es una tarea que necesita el tiempo y la distancia que toma un vehículo en pasar de un lugar a otro. Para esto el sistema necesita ser configurado con un punto de entrada y un punto de salida en el eje X. A partir de ahora a estos dos puntos se les llamará simplemente por punto A y punto B. Los puntos A y B deben ser colocados de manera que los vehículos deben pasar completamente por cada uno de ellos. Además, cada una de las muestras deben tener diferentes valores para los puntos A y B, sin embargo, para este caso no fue necesario implementar diferentes valores, ya que todos los videos fueron tomados en el mismo lugar. La Figura 4.5 muestra donde fueron colocados los puntos A y B en las muestras, denotados por dos líneas azules verticales.



Figura 4.5: Límites para lugar de las muestras.

Una vez que se deciden los valores para los puntos A y B, se genera el archivo csv. Para esto se toma en cuenta el segundo en que pasa debe ser lo más cercano posible al centroide del vehículo cuando pasa por el punto B. Es importante ingresar una descripción, aunque esta no va a ser usada por el sistema. Se volverá importante para validar que el vehículo al que se le tomó la velocidad es el mismo que detectó el sistema.

4.2. Obtención de velocidad

Como se mencionó anteriormente en la Sección 4 la obtención de la velocidad se divide en tres partes las cuales están representadas en el Figura 4.6 y se describen a continuación.

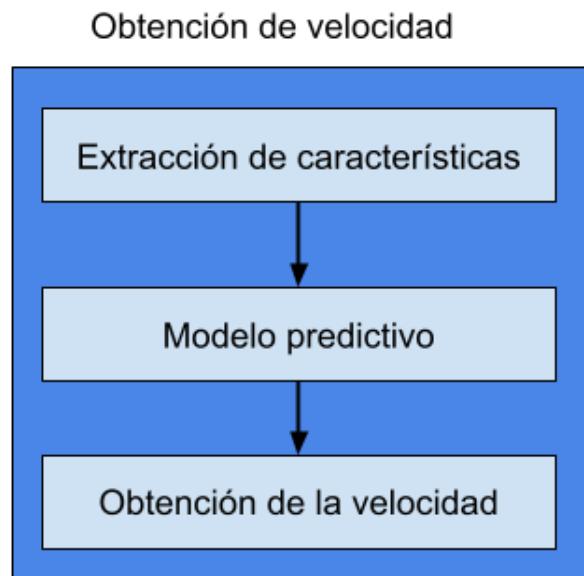


Figura 4.6: Proceso de obtención de velocidad.

4.2.1. Extracción de características

En el paso anterior para cada muestra se creo un archivo csv el cual servirá de entrada para el paso actual.

El sistema se encarga de leer el video utilizando la biblioteca OpenCV con la cual se extraen los fotogramas del video, cada fotograma pasa por la red neuronal YOLO la cual se encarga de identificar todos los vehículos que aparecen en el fotograma. YOLO regresa las coordenadas de cada objeto detectado dentro del fotograma, estas coordenadas son usadas para dibujar un recuadro para cada objeto. La Figura 4.7 muestra dos vehículos detectados.



Figura 4.7: Detección de vehículos dentro de recuadros.

El seguimiento de los vehículos se realiza por medio del Filtro Kalman el cual determina su ubicación en el próximo fotograma. El sistema se encarga de guardar todas las ubicaciones de los vehículos en el transcurso del tiempo siendo capaz dibujar todo el trayecto que han tenido cada uno de ellos. Sumando el uso de regresión lineal se crea una recta que corresponde a las trayectorias de los vehículos. La Figura 4.8 muestra un vehículo detectado en un recuadro blanco, una línea amarilla con la trayectoria del vehículo y una línea roja la cual es generada con regresión a partir de la trayectoria del vehículo.



Figura 4.8: Detección de vehículos en punto B, sus trayectorias y triangulación del generado con el punto A y B.

Otra característica que se identifica en la Figura 4.8 es la creación de un triángulo en color negro, con el cual se identifica el ángulo detectado para el vehículo.

Cando el sistema detecta que un vehículo pasa por el punto A, este guarda la información del estado del vehículo (Figura 4.9). Puede haber múltiples vehículos pasando en ese momento y todos serán detectados por el sistema. El vehículo del cual se obtienen sus características esta identificado con un recuadro de color amarillo, mientras que el resto de los vehículos están dentro de un recuadro color amarillo.



Figura 4.9: Punto A con vehículo en recuadro blanco.

Aunque se guardó el estado del vehículo cuando paso por el punto A, este no genera una línea para el archivo csv resultante. Es hasta que el vehículo pasa por el punto B y coincide con los segundos en el archivo csv de entrada que el sistema guarda un nuevo dato en el archivo de salida. Los vehículos que no cuentan con una línea en el archivo csv de entrada no se les guarda su información. La Figura 4.10 muestra cuando el vehículo detectado en el punto A pasa por el punto B.

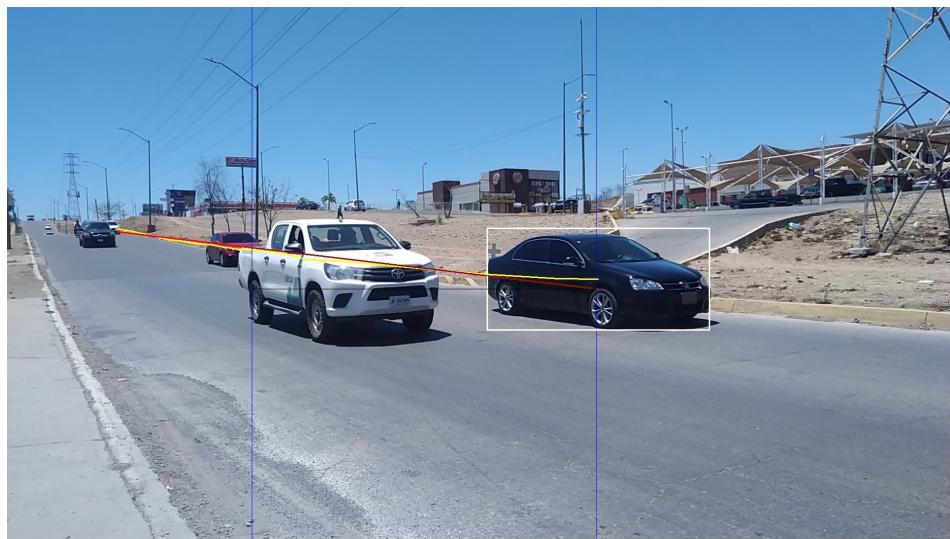


Figura 4.10: Punto B con vehículo en recuadro blanco.

La Tabla 4.1 muestra las características más importante que son extraídas por el sistema y una descripción.

Tabla 4.1: Características obtenidas por el sistema.

Característica	Descripción
Ángulo Salida	Ángulo a partir del punto de entrada hasta el punto de salida
Distancia de salida	Distancia recorrida desde el punto de entrada hasta el punto de salida
Área Entrada	Área detectada del vehículo en pixeles en el punto de entrada
Área Salida	Área detectada del vehículo en pixeles en el punto de salida
FPS	Fotogramas por segundo del video
Tiempo	Tiempo que le tomo al vehículo para pasar del punto entrada al de salida
Velocidad	Velocidad detectada por el radar
Carril	Carril por que pasa el vehículo
Identificador	Identificador correspondiente a una imagen generada

El sistema además de generar un archivo csv, crea una imagen de salida la cual corresponde a una línea del archivo resultante, esta imagen está formada por dos imágenes una al lado de la otra, la imagen de la izquierda representa el vehículo cuando entra en el punto A y la imagen de la derecha es cuando el vehículo pasa por el punto B. La Figura 4.11 muestra un ejemplo de esta imagen de salida.



Figura 4.11: Imagen resultado para cada línea de archivo csv.

Una vez que el sistema genera las imágenes de salida. El archivo csv resultante valida que el vehículo detectado sea el correcto y que el recuadro blanco que define la detección del vehículo contenga la mayor parte del vehículo.

Para el caso de validar que el vehículo sea el correcto, es necesario ver cada una de las imágenes generadas y leer la descripción en el archivo csv de entrada. En caso de identificar un vehículo que no corresponda, se debe modificar los segundos en el archivo csv de entrada de tal manera que el sistema detecte el vehículo correcto, esto implica ejecutar el sistema nuevamente para generar los datos de nuevo.

Existen dos casos en los que el sistema no podrá identificar el vehículo. El primer caso implica que el vehículo sea obstaculizado por otro. Por lo cual, la detección será del vehículo que esta frente al vehículo que se le esta realizando el seguimiento. El segundo caso corresponde cuando al vehículo no se le realizo el seguimiento completo del punto A al B de forma correcta, esto implica que el vehículo no fue detectado en el camino. Para estos dos caso se elimina la línea correspondiente en el archivo de entrada, lo cual también implica ejecutar el sistema nuevamente para generar los datos nuevamente.

Para validar que el área de detección del vehículo sea completa, se realizó una inspección visual de las imágenes generadas. Durante la inspección se identifica el vehículo cuando entra en el punto A (Figura 4.12) el cual debe ser detectado completamente por el recuadro blanco, al mismo tiempo que se identifica el vehículo cuando pasa por el punto B (Figura 4.13) el cual también debe ser identificado completamente por el recuadro blanco.



Figura 4.12: Imagen válida con vehículo entrando representando una línea del archivo csv.



Figura 4.13: Imagen válida con vehículo saliendo representando una línea del archivo csv.

Por otra parte, hay ocasiones en la cuales el sistema solo detecta parte del vehículo. En este caso la muestra se considera como inválida. Un ejemplo, la Figura 4.14 muestra el vehículo detectado completamente en el punto A, mientras que cuando el vehículo sale por el punto B es detectado solo una parte (Figura 4.15).

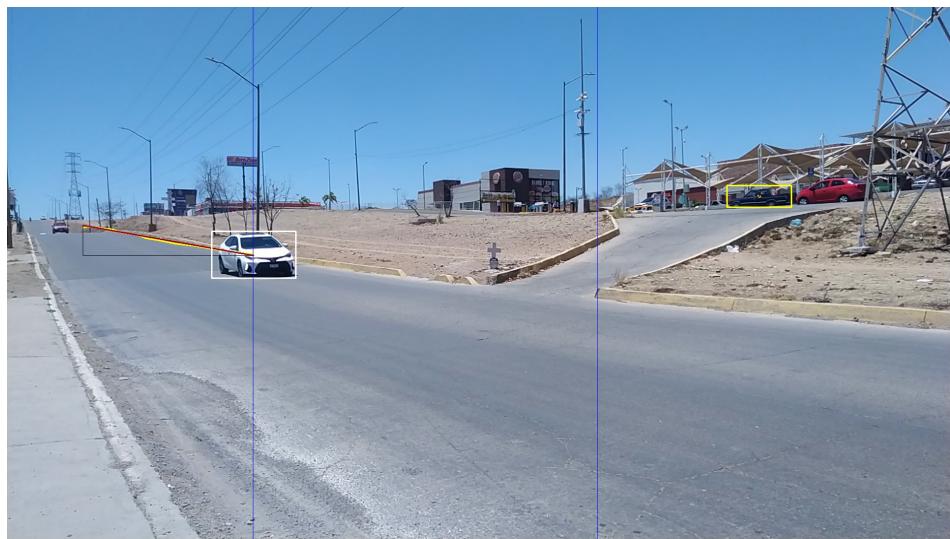


Figura 4.14: Imagen inválida (parte izquierda) con vehículo entrando representando una línea del archivo csv.



Figura 4.15: Imagen inválida (parte derecha) con vehículo saliendo representando una línea del archivo csv.

Una vez que se identifica una imagen inválida se elimina la muestra correspondiente en el archivo csv de salida. Esta muestra se reconoce por el identificador descrito en la Tabla 4.1. Para este caso no es necesario volver a ejecutar el sistema.

4.2.2. Modelo predictivo

En esta sección se describe la metodología utilizada para crear un proceso o modelo el cual es utilizado para determinar la velocidad de vehículos. Se implementaron dos procesos para esta tarea. El primer proceso utiliza la fórmula física de la velocidad, a partir de esta fórmula se busca un factor de correlación entre la distancia en píxeles recorrida por el vehículo y la distancia en metros. Mientras que el segundo proceso implementado usa un modelo de aprendizaje máquina utilizando bibliotecas de Python las cuales ayudan a crear un modelo predictivo con el cual se determina la velocidad del vehículo.

4.2.2.1. Estimar velocidad basado en factor de correlación

Para determinar la velocidad utilizando el conjunto de datos generado por el sistema se inicia utilizando la fórmula física descrita en la Ecuación 4.1 para el cálculo de la velocidad.

$$Velocidad = \frac{Distancia}{Tiempo} \quad (4.1)$$

En este caso las muestras fueron tomadas por un Radar que solo proporciona la velocidad en Kilómetros o en Millas, así que necesitamos convertir este valor a metros sobre segundo, utilizando la Ecuación 4.2.

$$\frac{M}{S} = \frac{18}{5} \times \frac{K}{H} \quad (4.2)$$

A partir de la Ecuación 4.1 donde se realiza cálculo de la velocidad y la conversión de la velocidad en metros por segundo en lugar de kilómetros por hora. La Figura 4.16 define los pasos que se deben seguir para modelar los datos obtenidos utilizando un factor de correlación. Se define el proceso para crear el modelo en el diagrama de flujo que se muestra en la Figura 4.16.

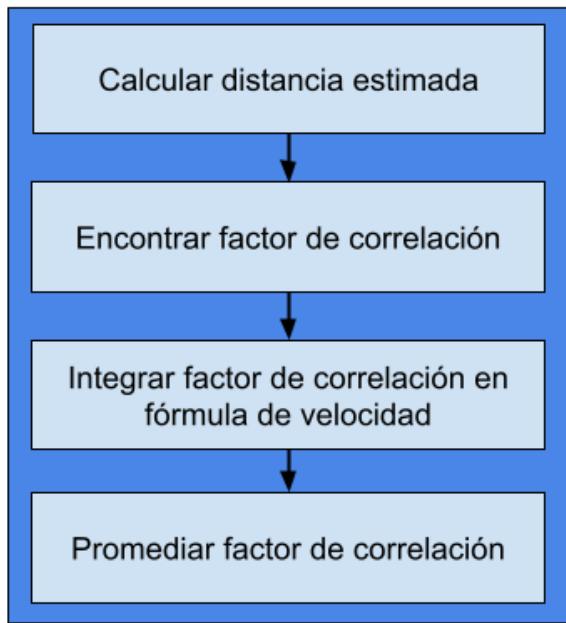


Figura 4.16: Diagrama de flujo para la creación de modelo con la fórmula física de la velocidad.

La Tabla 4.1 muestra que los tres parámetros necesarios para estimar la velocidad, sin embargo, el valor de la distancia está en pixeles, por lo que es necesario convertir el valor a metros. Este valor es convertido con la Ecuación 4.1 despejando la distancia, obteniendo la Ecuación 4.3.

$$Distancia\ Estimada = Tiempo \times Velocidad \quad (4.3)$$

Una vez que se tiene la distancia estimada, se busca encontrar la correlación entre ésta y la distancia en pixeles, con lo cual se propone la Ecuación 4.4.

$$B = \frac{Distancia\ Estimada}{Distancia\ Pixeles} \quad (4.4)$$

A la constante de correlación se le llamará B, Donde B es el valor utilizado para calcular una velocidad estimada con respecto a la distancia y el tiempo. Por lo que al ajustar la Ecuación 4.1 con la constante de correlación (Ecuación 4.4) tenemos la Ecuación 4.5.

$$Velocidad = \frac{Distancia\ Pixeles}{Tiempo} \times B \quad (4.5)$$

Sin embargo, es necesario encontrar la correlación de B en todas las muestras. Por lo cual se propone encontrar un valor de B que modele la gran mayoría de muestras. Para esto se calculó el valor medio de B definido por la Ecuación 4.6.

$$\bar{B} = \frac{\sum B}{n} \quad (4.6)$$

Una vez se tiene el promedio B (\bar{B}) se puede sustituir por B en la Ecuación 4.5 lo cual quedaría como muestra la Ecuación 4.7.

$$Velocidad = \frac{Distancia\ Pixeles}{Tiempo} \times \bar{B} \quad (4.7)$$

4.2.2.2. Estimar velocidad utilizando modelos de regresión

El proceso para inferir un dato (en este caso la velocidad) tanto para un modelo de regresión o de aprendizaje maquina es el mismo proceso. Este proceso inicia separando la información (en conjuntos de entrenamiento y validación), entrenando (refinando) el modelo predictivo, para finalmente estimar la velocidad (validar el modelo). La Figura describe las tres etapas descritas para la obtencion de la velocidad.

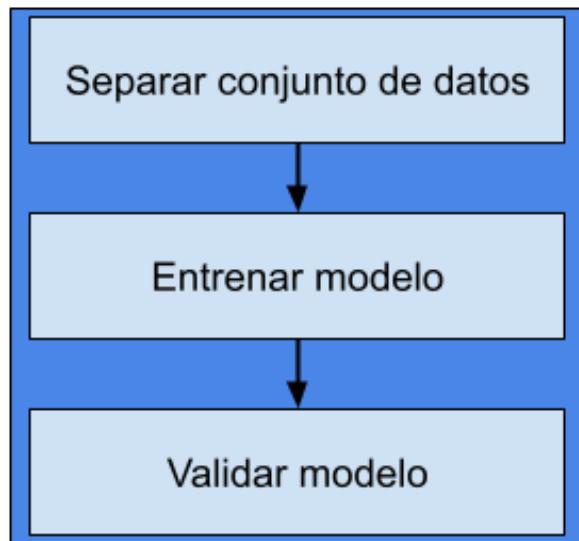


Figura 4.17: Diagrama de flujo para definir un modelo predictivo de la velocidad.

El primer paso es separar el conjunto de datos en un conjunto de entrenamiento y valida-

ción. Normalmente esto se realiza en un porcentaje de 70 % (entrenamiento) - 30 % (validación). Una vez separado el conjunto de datos se procede a la optimización de los parámetros del modelo o también llamado entrenamiento. Por último, se utiliza el modelo con el conjunto de validación para validar que los resultados obtenidos.

4.2.3. Obtención de la velocidad

Para la obtención de la velocidad se parte de las características extraídas de las secuencias de imágenes y la Ecuación 4.7 o el modelo generado en la Sección 4.2.2.1.

Por ejemplo, se calcula el valor de B promedio el cual es igual a 0.0162, el sistema obtiene la distancia en pixeles y el tiempo que le tomo al vehículo pasar del punto A al punto B, los cuales son igual a 962 y 0.933 respectivamente, ahora solo es cuestión de sustituir estos valores para conseguir la velocidad estimada.

$$\text{Velocidad} = \frac{962}{0.933} \times 0.0162 = 16.703 \text{ m/s} \quad (4.8)$$

No obstante, esta estimación está en M/S, entonces se necesita convertirlo a K/H usando la Ecuación 4.9.

$$16.703 \text{ m/s} \times \frac{18}{5} = 60.13 \text{ k/h} \quad (4.9)$$

Este es el resultado de estimar la velocidad de un vehículo utilizando el método propuesto, este es un método simple el cual ayudó como punto de referencia para futuros experimentos, por ejemplo, usando bibliotecas como Scikit para utilizar modelos ya existentes en esta biblioteca o Tensor Flow para crear una red neuronal con las características que mejor se adecuen a este caso.

Capítulo 5

Resultados

En esta sección se presentan los resultados de implementar el sistema con el cual se obtuvo un conjunto de datos para realizar experimentos con este y calcular la velocidad vehicular.

5.1. Lugar para toma de muestras

Para la toma de las muestras se eligió un lugar de la ciudad donde el flujo de vehículos es considerablemente constante. Este lugar fue en la calzada las Torres frente a Plaza San Isidro en la ciudad de Culiacán, Sinaloa. Una parte de las muestras fueron tomadas en las mañanas entre las 10 y las 12 hrs. Mientras que otra parte de las muestras fueron tomadas entre las 5 y 7 P.M. Las muestras indicaron que por la tarde se tuvo un mayor flujo vehicular, ya que en algunos casos de las muestras hay hasta 30 segundos con ausencia de vehículos en las muestras tomadas por la mañana. La Figura 5.1 es del lugar donde se tomaron las muestras.



Figura 5.1: Lugar donde fueron tomadas las muestras.

Este lugar tiene un flujo de oeste a este, lo cual para la cámara es un flujo de derecha a izquierda. También se aprecia que la calle por la cual pasan los vehículos no está nivelada, para la vista de la cámara los vehículos pasan de arriba hacia abajo. Como se mencionó en la Metodología, hubo especial cuidado a la hora de posicionar la cámara para que los vehículos no pasen paralelamente a la vista de la misma, por lo que se posicionó apuntando un poco hacia la subida de la calle.

5.2. Muestras obtenidas

La implementación del sistema da como resultado un total de 178 videos tomados, de los cuales solo fueron procesados un total de 29 videos, la baja cantidad de videos procesados fue debido al tiempo de ejecución del sistema, además de que en algunos casos había que volver a procesar un mismo video varias veces, el sistema procesa entre 30 y 45 fotogramas por segundo lo cual se traduce en el mejor de los casos en 1.333 segundos procesar un segundo real de video.

Los 29 videos procesados dieron un total de 532 muestras, con datos se realizaron los experimentos. Para los experimentos los datos fueron separados por carril 0, 1 y 2, los cuales representan el primer carril, carril central y último carril. Los datos para cada carril son los

siguientes:

- Primer carril: 8 datos (Figura 5.2)
- Carril central: 239 datos (Figura 5.3)
- Último carril: 285 datos (Figura 5.4)

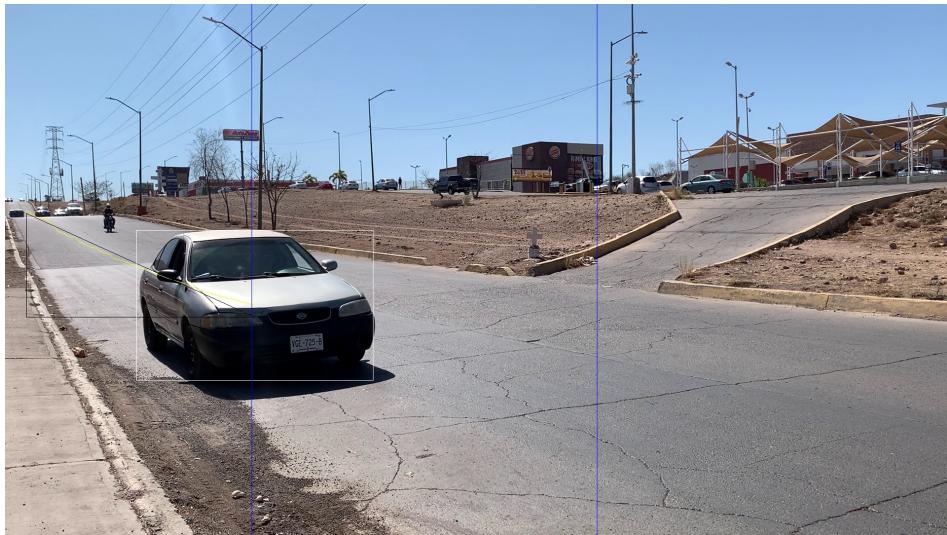


Figura 5.2: Muestra de primer carril.



Figura 5.3: Muestra de segundo carril.



Figura 5.4: Muestra de último carril.

Como se nota la diferencia del número de datos para el primer carril es considerablemente bajo con respecto a los otros dos carriles, por lo cual se omiten los experimentos para este carril, dejando solamente el cerril central y el último carril.

5.3. Hardware utilizado

El hardware utilizado fue una computadora física personal con las siguientes características:

- CPU: Intel® Core™ i3-8100 CPU
- RAM: 32GB DDR4
- GPU: GeForce RTX 2070 SUPER
- OS: Ubuntu 20.04.2 LTS

5.4. Experimentos

Principalmente se implementan 3 experimentos, el primero es utilizando el cálculo mostrado en la Sección 4.2.2, el segundo es utilizando modelos implementados en la biblioteca

Scikit Learn de Python y por último con el uso de Tensor Flow implementado una red neuronal de capas densas Figura 5.5.

En el caso de Scikit y Tensor Flow, se debe considerar que para cada entrenamiento se pueden obtener diferentes resultados. Esto debido a que los valores iniciales para cada modelo son aleatorios, entonces es posible tener diferentes precisiones entre un entrenamiento y otro, a pesar de que se utilice el mismo conjunto de datos. Por esta razón, el entrenamiento se realizó un número considerable de veces con el conjunto de datos creado, con lo cual se tienen diferentes resultados cada vez. Se presenta el valor máximo de error obtenido con respecto al total de veces que se repitieron los entrenamientos, el valor mínimo, el promedio y la desviación estándar de todos los entrenamientos.

Para todos los casos fue implementada la métrica de evaluación del Error Absoluto Medio, lo cual significa que las predicciones tienen en promedio un margen de error de N M/S o su equivalente en K/H, con esto proporciona una idea de que tan precisa es la predicción con los diferentes modelos implementados.

5.5. Regresión lineal para cálculo de la velocidad

En la Sección 4.2.2 se habla sobre el cálculo de la velocidad utilizando una regresión simple implementada utilizando solamente la fórmula básica de la velocidad, en estos experimentos se utilizó esta misma fórmula para el conjunto de datos creado con el sistema, utilizando solamente el carril central y el último carril.

5.6. Resultados Regresión lineal para cálculo de la velocidad

En la Tabla 5.1 de resultados se aprecia que los mejores resultados están en el último carril con 5.86 K/H, mientras que el carril central tiene 7.74 K/H. Se toman estos resultados como punto de referencia para mejorar las futuras predicciones, ya que estos no son tan buenos en comparación por ejemplo del margen de error del radar Bushnell utilizado.

Tabla 5.1: Resultados usando regresión simple

	M/S	K/H
Carril Central	2.15	7.74
Último Carril	1.63	5.86

5.7. Modelos de Regresión

Para el caso de los modelos de regresión, la implementación fue realizada utilizando la librería de Scikit-Learn para implementar los siguientes modelos:

- Regresión Lineal (RL)
- Regresión de crestas (RC)
- Lasso (Operador de Selección y Contracción Mínima Absoluta - Lasso)
- Regresión de la cresta bayesiana (RCB)
- Regresión Elastic Net (Elastic Net)
- Regresión Random Forest (RRF)
- Regresión de Máquina de Soporte Vectorial (RSVM)

Para los experimentos usando la biblioteca Scikit se realizó el entrenamiento de los modelos un total de 800 veces, ya que esta es la cantidad máxima permitida por los recursos del hardware utilizado, con estos resultados se obtiene el valor mínimo, el valor máximo y el promedio de todos los experimentos.

5.7.1. Carril central

La Tabla 5.2 muestra los resultados para el carril central en kilómetros por hora, el valor mínimo obtenido es para la Regresión Lineal, pero hay promedios cercanos en milésimas para la Regresión Lineal (RL), Regresión Ridge (RC), Lasso (Lasso), Regresión Bayesiana (RCB) y Elastic Net (Elastic net).

Tabla 5.2: Resultados para carril central

	RL	RC	Lasso	RCB	Elastic Net	RRF	RSVM
MIN	1.246	1.256	1.271	1.252	1.278	1.434	1.597
MAX	2.158	2.140	2.148	2.153	2.155	2.282	2.441
D.Est.	0.143	0.142	0.143	0.143	0.143	0.141	0.163
Prom.	1.694	1.694	1.699	1.695	1.701	1.828	1.960

5.7.2. Último Carril

La Tabla 5.3 muestra los resultados para el último carril en kilómetros por hora, en este caso los valores mínimos iguales para los modelos de Regresión Lineal (RL), Regresión Ridge (RC), Lasso (Lasso) y Regresión Bayesiana (RCB). Mientras que para los promedios los valores más bajos son para la Regresión Lineal (RL) y la Regresión Bayesiana (RCB).

Tabla 5.3: Resultados para último carril

	RL	RC	Lasso	RCB	Elastic Net	RRF	RSVM
MIN	0.738	0.737	0.739	0.738	0.758	0.841	1.514
MAX	1.222	1.223	1.233	1.222	1.278	1.400	2.576
D. Est.	0.081	0.081	0.082	0.081	0.084	0.094	0.141
Prom.	0.956	0.965	0.975	0.956	0.997	1.087	1.939

5.8. Modelo Neuronal

Para la implementación de la red neuronal se utilizó la biblioteca de Tensor Flow, donde se utilizó una red neuronal simple con 2 neuronas de entrada, 5 neuronas en la capa oculta y 1 neurona en la capa de salida. La Figura 5.5 muestra esta red.

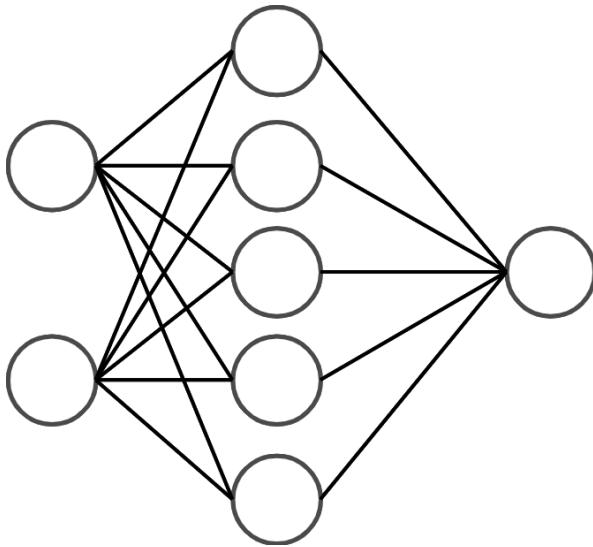


Figura 5.5: Red neuronal implementada en Tensor Flow.

Para la implementación con la biblioteca Tensor Flow se realizaron 1000 entrenamientos, ya que esta es la cantidad máxima permitida por los recursos del hardware utilizado, con los resultados de estos entrenamientos se obtuvo el valor mínimo el valor máximo y el promedio de todos los experimentos.

5.8.1. Carril central

La Tabla 5.4 muestra los resultados para el carril central en kilómetros por hora, se nota que para estos experimentos el promedio es de 12.762 K/H con un valor mínimo de 1.872 K/H.

Tabla 5.4: Resultados utilizando Tensor Flow carril central

	Error
MIN	1.872
D. Est.	13.397
Prom.	12.767

5.8.2. Último carril

La Tabla 5.5 muestra los resultados para el último carril en kilómetros por hora, la cual muestra un valor mínimo de 2.128 K/H con un promedio de 13.172 K/H.

Tabla 5.5: Resultados utilizando Tensor Flow último carril

	Error
MIN	2.127
D. Est.	11.843
Prom.	13.174

5.9. Análisis de resultados

Los mejores resultados con el uso de los modelos con la biblioteca Scikit principalmente en el último carril, esto debido a que es el carril con la mayoría de las muestras. Por otra parte, en el caso de los experimentos utilizando el método del factor de correlación hay buenos resultados, al final está el modelo de Tensor Flow.

En conclusión el caso del comparativo entre el uso de la biblioteca de Scikit y el factor de correlación, hay una falta de datos para aumentar la precisión de las inferencias, esto también se observa en el caso del uso de Tensor Flow, sin embargo, a este modelo hay que agregarle que es un modelo simple el cual solo usa capas densas, con lo cual mejoraría aumentando el número de neuronas, el uso de otras arquitecturas como las redes neuronales convolucionales entre otros.

Capítulo 6

Conclusiones

En esta sección se presentan las conclusiones de la implementación de la metodología propuesta y sus resultados, así como las aportaciones y el trabajo futuro.

6.1. Conclusión

Con el uso de la metodología propuesta en este trabajo se creó un sistema para obtener un conjunto de datos, el cual sirve como base para realizar experimentos con diferentes metodologías ya sean estadísticas o de inteligencia artificial para determinar la velocidad de vehículos u otros objetos, con la capacidad de ser adaptando a las necesidades o integrando más componentes para obtener otros parámetros que se puedan considerar de importancia.

El sistema muestra que es relativamente sencillo obtener una gran cantidad de parámetros de un video implementando algoritmos existentes de flujo óptico y modelos ya entrenados de aprendizaje máquina. Al mismo tiempo se puede comparar una metodología simple para calcular la velocidad con otros métodos más sofisticados como los modelos implementados con la biblioteca Scikit o la simple red neuronal implementada con la biblioteca Tensor Flow.

Sin embargo, uno de los mayores retos para la implementación de este sistema ha sido la toma de muestras, ya que se tiene que encontrar un lugar ideal donde haya suficiente flujo vehicular para obtener la mayor cantidad de datos posible, a esto hay que sumarle que en caso de integrar nuevos componentes las muestras deben ser tomadas nuevamente.

Otro punto importante que se debe mencionar es el tiempo de procesamiento de cada muestra, como se presentó anteriormente en el peor de los casos puede tomar hasta un 33 %

más tiempo del que dura el video original, además de la necesidad de que una persona sea quien limpие y valide las muestras, sumando más tiempo al procesamiento de cada video.

Los resultados obtenidos con el conjunto de datos generado ofrecen una idea de los diferentes métodos que podemos implementar y obtener resultados aceptables, pudiendo mejorar aumentando el tamaño del conjunto de datos, cambiando o mejorando el modelo implementado, puesto que en el mejor de los casos se tiene un error absoluto promedio de 0.205 M/S lo que equivale a 0.738 K/H siendo esta una mejor precisión a la del radar utilizado. Los métodos de aprendizaje profundo han demostrado en distintas áreas que son capaces de aprender patrones con gran precisión, pero en el caso de los experimentos realizados, el aprendizaje profundo no dispuso de tan buenos resultados, esto debido a la sencillez de la red neuronal utilizada.

6.2. Aportaciones

Este proyecto sirve como punto de partida o base para futuros proyectos relacionados con el análisis de tráfico vehicular, ya que en la investigación realizada del estado del arte se encontraron trabajos muy prometedores, pero la mayor parte de estos trabajos no cuentan con el código fuente para replicar los experimentos realizados o este es obsoleto.

Tomando este proyecto como base se puede comenzar a estudiar cómo trabaja un sistema para obtener un conjunto de datos y comenzar a trabajar en una extensión de este, obteniendo avances más rápido que si lo hicieramos solamente investigando proyectos que no cuentan con el código fuente.

El conjunto de datos generado también ha sido un importante aporte dado que la toma de las muestras y la obtención del conjunto de datos a partir de las mismas, ha sido una de las tareas a las que se les invirtió más tiempo, adicionalmente las muestras pueden ser utilizadas nuevamente para generar el conjunto de datos con más parámetros de ser necesario.

6.3. Trabajo futuro

El proyecto tiene diversos puntos de mejoras que pueden ser implementados más adelante. Uno de estos es eliminar la necesidad de una persona que se encargue de realizar la limpieza y validación del conjunto de datos generado en vista de que, esta es la tarea que toma más tiempo, por lo cual podemos hacer que sea el mismo sistema el que se valide a sí mismo entrenando una red neuronal que se encargue o simplemente diseñar un dispositivo que nos proporcione los datos que se requieren directamente.

Otra actividad de mejora que se puede aplicar al sistema es la integración con cámaras RGBD, con las cuales se obtiene información sobre la profundidad de la escena con lo cual nos proporciona más parámetros que podrían ayudar a mejorar la precisión de los modelos entrenados, no obstante, esto implicaría volver a tomar las muestras utilizando esta cámara.

Bibliografía

- 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 Conference*, 1-6. <https://doi.org/10.1109/TENCON.2015.7373162> (pages 36, 40, 42)
- Bell, D., Xiao, W. & James, P. (2020). Accurate Vehicle Speed Estimation from Monocular Camera Footage. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020, 419-426. <https://doi.org/10.5194/isprs-annals-V-2-2020-419-2020> (pages 39, 41, 42)
- Bevilacqua, M., Tsourdos, A. & Starr, A. (2016). Egomotion estimation for monocular camera visual odometer. *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, 1-6. <https://doi.org/10.1109/I2MTC.2016.7520579> (pages 35, 40, 42)
- Buduma, N. (2017). *Fundamentals of Deep Learning Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly Media, Inc. (Page 22).
- 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 41, 42)

Buy velocity speed gun and more. (s.f.). <https://www.bushnell.com/additional-products/speed-guns/velocity-speed-gun/BU-101911.html>. (Page 45)

Carro-Pérez, E. H. & Ampudia-Rueda, A. (2019). Conductas de riesgo al conducir un automóvil en zonas urbanas del sur de Tamaulipas y la Ciudad de México. *CienciaUAT*, 13(2), 100-112 (page 1).

CIFAR-10. (s.f.). <https://www.cs.toronto.edu/~kriz/cifar.html>. (Page 13)

Dong, H., Wen, M. & Yang, Z. (2019). Vehicle Speed Estimation Based on 3D ConvNets and Non-Local Blocks. *Future Internet*, 11, 123. <https://doi.org/10.3390/FI11060123>
(pages 38, 41, 42)

Fernández-Llorca, D., Salinas, C., Jimenez, M., Morcillo, A., Izquierdo, R., Lorenzo Díaz, J. & Sotelo, M.-A. (2016). Two-camera based accurate vehicle speed measurement using average speed at a fixed point, 2533-2538. <https://doi.org/10.1109/ITSC.2016.7795963> (pages 34, 40, 42)

Flach, P. (2012). *Machine Learning The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press. (Page 20).

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly. (Page 15).

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. <https://doi.org/10.2200/S00762ED1V01Y201703HLT037>. (Page 28)

Goyal, P., Pandey, S. & Jain, K. (2018). *Deep Learning for Natural Language Processing Creating Neural Networks with Python*. Palash Goyal. (Pages 20, 21).

- Harrington, P. (2012). *Machine Learning in Action*. Manning Publications. (Page 11).
- Impedovo, D., Balducci, F., Dentamaro, V. & Pirlo, G. (2019). Vehicular traffic congestion classification by visual features and deep learning approaches: a comparison. *Sensors*, 19(23), 5213 (page 2).
- Jalalat, M., Nejati, M. & Majidi, A. (2016). Vehicle detection and speed estimation using cascade classifier and sub-pixel stereo matching. *2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS)*, 1-5. <https://doi.org/10.1109/ICSPIS.2016.7869890> (pages 33, 40, 42)
- Kamoji, S., Koshti, D., Dmonte, A., George, S. & Pereira, C. (2020). Image Processing based Vehicle Identification and Speed Measurement, 523-527. <https://doi.org/10.1109/ICICT48043.2020.9112419> (pages 36, 40, 42)
- Kurniawan, A., Ramadlan, A. & Yuniarno, E. M. (2018). Speed Monitoring for Multiple Vehicle Using Closed Circuit Television (CCTV) Camera. *2018 International Conference on Computer Engineering, Network and Intelligent Multimedia (CENIM)*, 88-93. <https://doi.org/10.1109/CENIM.2018.8710854> (pages 33, 40, 42)
- Lee, K.-H. (2021). A Study on Distance Measurement Module for Driving Vehicle Velocity Estimation in Multi-Lanes Using Drones. *Applied Sciences*, 11(9). <https://doi.org/10.3390/app11093884> (pages 36, 40, 42)
- Li, S., Yu, H., Zhang, J., Yang, K. & Bin, R. (2014). Video-based traffic data collection system for multiple vehicle types. *IET Intelligent Transport Systems*, 8(2), 164-174. <https://doi.org/https://doi.org/10.1049/iet-its.2012.0099> (pages 33, 40, 42)
- Loor, C. (2017). Visual Speedometer: Learning Velocity from Two Images. *University of Amsterdam* (pages 39, 41, 42).

- Luvizon, D., Nassu, B. & Minetto, R. (2014). Vehicle speed estimation by license plate detection and tracking. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. <https://doi.org/10.1109/ICASSP.2014.6854869> (pages 34, 40, 42)
- Mcculloch, W. & Pitts, W. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5 (page 10).
- Minsky, M. & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press. (Pages 10, 20).
- Moritz Kampelmuhler, M. G. M. & Feichtenhofer, C. (2018). Camera-based vehicle velocity estimation from monocular video. *Institute of Electrical Measurement and Measurement Signal Processing* (pages 37, 41, 42).
- Overview. (s.f.). <https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/overview.html>. (Page 31)
- Partida, M. V. (2020). *Sistema de diseño y optimización de redesconvolucionales por razonamiento bayesianoutilizando entrenamientos parciales y distribuidos* (Tesis doctoral). Instituto Tecnológico de Culiacán. (Page 24).
- Patterson, J. & Gibson, A. (2017). *Deep learning: A practitioner's approach*. .o'Reilly Media, Inc.” (Pages 11, 14, 25).
- Peng, Y., Liu, X., Shen, C., Huang, H., Zhao, D., Cao, H. & Guo, X. (2019). An Improved Optical Flow Algorithm Based on Mask-R-CNN and K-Means for Velocity Calculation. *Multidisciplinary Digital Publishing Institute*. <https://doi.org/10.3390/app9142808> (page 39)

- Rao, Q. & Frtuník, J. (2018). Deep learning for self-driving cars: Chances and challenges. *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, 35-38 (page 2).
- Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788. <https://doi.org/10.1109/CVPR.2016.91> (pages 37, 41, 42)
- Rosebrock, D. A. (2017). *Deep Learning for Computer Vision with Python*. PYIMAGE SEARCH. (Pages 6, 11, 17, 19, 20, 25).
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6). <https://doi.org/10.1037/h0042519> (pages 10, 20)
- Rumelhart, D. E. & McClelland, J. L. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press. (Pages 10, 21, 22).
- Schoepflin, T. & Dailey, D. (2003). Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2), 90-98. <https://doi.org/10.1109/TITS.2003.821213> (pages 35, 40, 42)
- Singh, S. (2015). Estimating Speed, Velocity, Acceleration and Angle Using Image Addition Method. *International Journal of Innovative Research in Computer and Communication Engineering*, 3, 11585-11589. <https://doi.org/10.15680/IJIRCCE.2015> (pages 32, 40, 42)

- Song, Z., Luand, J., Zhang, T. & Li, H. (2020). End-to-end Learning for Inter-Vehicle Distance and Relative Velocity Estimation in ADAS with a Monocular Camera. *Cornell University* (pages 38, 41, 42).
- Swamynathan, M. (2017). *Mastering Machine Learning with Python in Six Steps A Practical Implementation Guide to Predictive Data Analytics Using Python*. Springer Science+Business Media New York. (Page 21).
- Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media. (Page 6).
- Vakili, E., Shoaran, M. & Sarmadi, M. (2020). Single-camera vehicle speed measurement using the geometry of the imaging system. *Multimedia Tools and Applications*, 79. <https://doi.org/10.1007/s11042-020-08761-5> (pages 35, 40, 42)
- van Rossum, G. (1994). Welcome to Python.org. <https://www.python.org/shell/>. (Page 29)
- Velázquez Narváez, Y., Zamorano González, B. & Ruíz Ramos, L. (2017). Siniestralidad vial en la frontera norte de Tamaulipas. Enfoque en los procesos administrativos de control. *Estudios fronterizos*, 18(36), 1-24 (page 1).
- Welch, G., Bishop, G. et al. (1995). An introduction to the Kalman filter. (Pages 8, 9).
- What is a container? (s.f.). <https://www.docker.com/resources/what-container>. (Page 30)
- Yang, L., Li, M., Song, X., Xiong, Z., Hou, C. & Qu, B. (2019). Vehicle Speed Measurement Based on Binocular Stereovision System. *IEEE Access*, 7, 106628-106641. <https://doi.org/10.1109/ACCESS.2019.2932120> (pages 34, 40, 42)
- Yang, L., Li, Q., Song, X., Cai, W., Hou, C. & Xiong, Z. (2021). An Improved Stereo Matching Algorithm for Vehicle Speed Measurement System Based on Spatial and Temporal Features. *IEEE Access*, 9, 106628-106641. <https://doi.org/10.1109/ACCESS.2021.3059120> (pages 34, 40, 42)

- poral Image Fusion. *Entropy*, 23(7). <https://doi.org/10.3390/e23070866> (pages 35, 40, 42)
- Yang, L., Luo, J., Song, X., Li, M., Wen, P. & Xiong, Z. (2021). Robust Vehicle Speed Measurement Based on Feature Information Fusion for Vehicle Multi-Characteristic Detection. *Entropy*, 23, 910. <https://doi.org/10.3390/e23070910> (pages 35, 40, 42)
- Yaqi Zhang, B. W. & Liu, W. (2017). Vehicle Motion Detection using CNN. *Stanford* (pages 37, 41, 42).
- Zaki, P. S., William, M. M., Soliman, B. K., Alexsan, K. G., Khalil, K. & El-Moursy, M. (2020). Traffic signs detection and recognition system using deep learning. *arXiv preprint arXiv:2003.03256* (page 1).