UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

IDENTIFICACIÓN AUTOMÁTICA DE VEHÍCULOS EMPLEANDO APRENDIZAJE PROFUNDO

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

FERNANDO CONTRERAS PÉREZ

BAJO LA DIRECCIÓN DE:

DR. OSCAR ALBERTO CHÁVEZ BOSQUEZ

EN CODIRECCIÓN:

DRA. BETANIA HERNÁNDEZ OCAÑA

CUNDUACÁN, TABASCO, A: OCTUBRE 2024

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

IDENTIFICACIÓN AUTOMÁTICA DE VEHÍCULOS EMPLEANDO APRENDIZAJE PROFUNDO

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

FERNANDO CONTRERAS PÉREZ

BAJO LA DIRECCIÓN DE:

DR. OSCAR ALBERTO CHÁVEZ BOSQUEZ

EN CODIRECCIÓN:

DRA. BETANIA HERNÁNDEZ OCAÑA

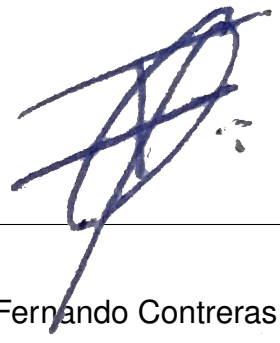
CUNDUACÁN, TABASCO, A: OCTUBRE 2024

Declaración de Autoría y Originalidad

En la Ciudad de Cunduacán el día siete del mes de octubre del año 2024, el que suscribe **Fernando Contreras Pérez**, alumno del Programa de la **Maestría en Ciencias de la Computación** con número de matrícula **212H13001**, adscrito a la **División Académica de Ciencias y Tecnologías de la Información**, de la Universidad Juárez Autónoma de Tabasco, como autor de la Tesis presentada para la obtención de Grado y maestría y titulada **Identificación automática de vehículos empleando aprendizaje profundo**, dirigida por el Dr. Oscar Alberto Chávez Bosquez y la Dra. Betania Hernández Ocaña.

DECLARO QUE: La Tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la LEY FEDERAL DEL DERECHO DE AUTOR (Decreto por el que se reforman y adicionan diversas disposiciones de la Ley Federal del Derecho de Autor del 01 de Julio de 2020 regularizándo, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita. Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad o contenido de la Tesis presentada de conformidad con el ordenamiento jurídico vigente.

Cunduacán, Tabasco a 07 de octubre de 2024.



Estudiante: Fernando Contreras Pérez

Cunduacán, Tabasco a 30 de septiembre de 2024

Oficio No. 1220/DACYTI/CP/2024

Asunto: Autorización de impresión de Tesis

C. Fernando Contreras Pérez
Egresado de la Maestría en Ciencias de la Computación

En virtud de que cumple satisfactoriamente los requisitos establecidos en el Reglamento General de Estudios de Posgrado vigente en la Universidad, informo a Usted que se autoriza la impresión del trabajo recepcional ***"Identificación automática de vehículos empleando aprendizaje profundo"***, para presentar examen y obtener el Grado de Maestro en Ciencias de la Computación.

Sin otro particular, aprovecho la ocasión para enviarle un afectuoso saludo.

Atentamente

MTE. Óscar Alberto González González
Director

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO
DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

C.c.p. Dr. Eddy Arquímedes García Alcocer. - Encargado del Despacho de la Coordinación de Posgrado DACYTI
Archivo.
Consecutivo.

M.T.E. OAGG/EAGA

Carretera Cunduacán-Jalpa Km. 1, Colonia Esmeralda, C.P. 86690.
Cunduacán, Tabasco, México.
Tel: (993) 358 1500 ext. 6727; (914) 336 0616; Fax: (914) 336 0870
E-mail: direccion.dacyti@ujat.mx

Carta de Cesión de Derechos

Villahermosa, Tabasco a 07 de octubre de 2024.

Por medio de la presente manifiesto haber colaborado como AUTOR en la producción, creación y/o realización de la obra denominada: **Identificación automática de vehículos empleando aprendizaje profundo**.

Con fundamento en el artículo 83 de la Ley Federal del Derecho de Autor y toda vez que, la creación y/o realización de la obra antes mencionada se realizó bajo la comisión de la Universidad Juárez Autónoma de Tabasco; entendemos y aceptamos el alcance del artículo en mención de que tenemos el derecho al reconocimiento como autores de la obra, y a la Universidad Juárez Autónoma de Tabasco mantendrá en un 100 % la titularidad de los derechos patrimoniales por un período de 20 años sobre la obra en la que colaboramos, por lo anterior, cedemos el derecho patrimonial exclusivo en favor de la Universidad.

COLABORADOR

Estudiante: Fernando Contreras Pérez

TESTIGOS

Dr. Oscar Alberto Chávez Bosquez

Dra. Betania Hernández Ocaña

Índice general

Tabla de contenido I

Índice de Figuras IV

Índice de Tablas VI

Resumen 1

Abstract 2

1. Generalidades 3

1.1. Introducción . . . 3

1.2. Planteamiento del problema . . . 5

1.2.1. Definición del problema . . . 5

1.2.2. Delimitación de la investigación . . . 6

1.3. Pregunta de investigación e hipótesis . . . 7

1.4. Objetivo general . . . 7

1.5. Objetivos específicos . . . 7

1.6. Justificación . . . 7

1.7. Metodología utilizada . . . 9

1.8. Descripción del documento . . . 12

2. Marco teórico 14

2.1. Marco conceptual . . . 14

2.1.1. Aprendizaje automático . . . 14

2.1.2. Aprendizaje profundo . . . 17
2.1.3. Redes neuronales multicapa . . . 19
2.1.4. Redes neuronales convolucionales . . . 24
2.1.5. Detección de objetos . . . 27
2.1.6. MobileNetSSD . . . 29
2.1.7. Seguimiento de objetos . . . 30
2.1.8. Métricas de rendimiento . . . 31
2.1.9. Detección automática de vehículos . . . 33
2.2. Marco referencial . . . 34
2.2.1. Sensores . . . 35
2.2.2. Visión computacional . . . 35
2.2.3. Aprendizaje profundo . . . 36
2.3. Marco tecnológico . . . 39

3. Sistema de detección, seguimiento y conteo de vehículos 41
3.1. Diagrama del sistema . . . 41
3.2. Integración del hardware . . . 42
3.3. Integración del software . . . 45
3.4. Configuración y puesta a punto . . . 47

4. Experimentos y Resultados 50
4.1. Diseño experimental . . . 52
4.1.1. Ubicación del sensor . . . 52
4.1.2. Configuración de parámetros . . . 53
4.1.3. Captura de datos . . . 55
4.2. Pruebas de detección . . . 55
4.3. Pruebas de *Tracking* . . . 61

5. Contribuciones, conclusiones y trabajos futuros 66
5.1. Conclusiones . . . 66
5.2. Trabajos futuros . . . 68

5.3. Creación del reporte estadístico 69

Anexo **70**

Bibliografía **71**

Índice de figuras

1.1. Ubicación de los espacios para estacionamiento del campus Chontalpa de la UJAT. 5
1.2. Diagrama de la metodología CRISP-DM (IBM-Corporation, 2021). 9
1.3. Diagrama de la metodología OSEMN (Lau, 2019). 10

2.1. Las neuronas artificiales fueron inspiradas por las neuronas biológicas (Elgendy, 2020). 19
2.2. Diagrama de un perceptrón (Elgendy, 2020). 20
2.3. Ejemplos de conjuntos de datos lineales (izquierda) y no lineales (derecha) (Elgendy, 2020). 21
2.4. Diagrama de una red neuronal con 3 perceptrones en una capa oculta (Elgendy, 2020). 22
2.5. Diagrama de una red neuronal con 6 capas ocultas con 6 neuronas cada una (Elgendy, 2020). 23
2.6. Diagrama de una red neuronal convolucional (Elgendy, 2020). 25

3.1. Diagrama integral del sistema. 42
3.2. Sensor OAK-D y Raspberry Pi 4 integrados. 44
3.3. Posiciones del sensor y ROI. 46
3.4. Sensor OAK-D y Raspberry Pi 4 integrados e instalados en su soporte. 49

4.1. Diagrama del diseño experimental. 51
4.2. Experimento principal en la entrada del campus Chontalpa de la UJAT usando la ROI y 3 posiciones diferentes del sensor OAK-D. 53

4.3. Ejemplos de detección de vehículos. 57
4.4. Ejemplo de conteo de vehíulos. 61

Índice de tablas

4.1. Detalle de las tres ubicaciones del sensor para la captura de los datos. 52
4.2. Configuración de los parámetros del sistema. 54
4.3. Configuración de parámetros seleccionada. 54
4.4. Detalle de las pruebas realizadas en los experimentos. 55
4.5. Métricas obtenidas por MobileNetSSD para la clase “Automóviles”. 58
4.6. Métricas obtenidas por MobileNetSSD para la clase “Motocicletas”. 58
4.7. Métricas obtenidas por MobileNetSSD para la clase “Autobuses”. 59
4.8. Métricas obtenidas por MobileNetSSD para la clase “Bicicletas”. 60
4.9. Resultados del conteo automático de la clase automóviles realizado con el algoritmo de *tracking*. 62
4.10. Resultados del conteo automático de la clase motocicletas realizado con el algoritmo de *tracking*. 63
4.11. Resultados del conteo automático de la clase autobuses realizado con el algoritmo de *tracking*. 64
4.12. Resultados del conteo automático de la clase bicicletas realizado con el algoritmo de *tracking*. 65

Resumen

Este trabajo de investigación se enfoca en la implementación y evaluación de un sistema de identificación, clasificación y conteo automático de vehículos utilizando tecnologías de Aprendizaje profundo. El principal objetivo de este sistema es el de mejorar el control de acceso vehicular en el campus Chontalpa de la Universidad Juárez Autónoma de Tabasco (UJAT). El sistema está compuesto por un sensor OAK-D y una Raspberry Pi para capturar, procesar y analizar los datos en tiempo real.

El sistema propuesto, basado en la red neuronal MobileNetSSD, ha logrado una identificación de vehículos con una precisión global de 0.81 y una sensibilidad de 0.91. Este desempeño en la identificación influye positivamente en la tarea de conteo automático.

El diseño experimental tomó en cuenta la recopilación y análisis de datos con diferentes configuraciones. Se ajustó el sistema a distintas condiciones de iluminación y climáticas. Las pruebas se realizaron en algunos puntos de la entrada del campus. También se variaron la ubicación y ángulos del OAK-D para mejorar la precisión de la detección. Este enfoque muestra que el sistema propuesto es adaptable a diferentes escenarios en condiciones naturales no controladas para entornos educativos. Todo esto, sin la necesidad de un equipo profesional costoso.

El sistema propuesto ha demostrado ser una solución funcional y económica para el control de acceso vehicular, mejorando la seguridad y la eficiencia en el uso de recursos. Para investigaciones futuras, se propone explorar otros modelos de detección como YOLO así como algoritmos de seguimiento más complejos basados en Aprendizaje profundo. Estas mejoras podrían aumentar la precisión y la sensibilidad del sistema.

Palabras clave: Inteligencia Artificial, Redes neuronales, Visión Computacional

Abstract

This research focuses on the implementation and evaluation of a system for automatic identification, classification and tracking of vehicles using Deep Learning. The main objective of this system is to improve vehicle access control on the Chontalpa campus of the Universidad Juárez Autónoma de Tabasco (UJAT). The system is composed of an OAK-D sensor and a Raspberry Pi to capture, process and analyze data in real time.

The proposed system, based on the MobileNetSSD neural network, has achieved vehicle identification with an overall precision of 0.81 and a sensitivity of 0.91. This identification performance positively influences the automatic tracking task.

The experimental design took into account the collection and analysis of data with different configurations. The system was adjusted to different lighting and weather conditions. The tests were carried out at several points at the entrance of the campus. The location and angles of the OAK-D were also varied to improve the detection accuracy. This approach shows that the proposed system is adaptable to different scenarios in uncontrolled natural conditions for educational environments. All this, without the need for expensive professional equipment.

The proposed system has proven to be a functional and economical solution for vehicle access control, improving security and efficiency in the use of resources. For future research, it is proposed to explore other detection models such as YOLO as well as more complex tracking algorithms based on Deep Learning. These improvements could increase the accuracy and sensitivity of the system.

Keywords: Artificial Intelligence, Neural Networks, Computer Vision

Capítulo 1

Generalidades

1.1. Introducción

La inseguridad es un problema social muy presente en México. Este problema provoca cambios en las costumbres cotidianas y genera impacto significativo tanto psicológico como económico. En el ámbito educativo, la situación es igualmente desfavorable. Según datos de las fiscalías estatales y las secretarías de Educación, México experimentó casi 7000 robos a escuelas entre marzo de 2020 y marzo de 2021 (Montes, 2021). A pesar de las medidas de seguridad, incluyendo la contratación de personal de vigilancia en las escuelas, particularmente en Tabasco, se siguen presentando robos. Además, el robo de autos ha experimentado un incremento notable, siendo este un delito que contribuye directamente a la economía criminal y está relacionado con otros delitos mayores (Ramírez-Sánchez, 2019; Tabasco Hoy, 2022).

Tomando en cuenta esta problemática, en este proyecto de investigación se propone un sistema de identificación automática de vehículos empleando Aprendizaje profundo, en específico redes neuronales convolucionales. Este sistema integra software y hardware especializado para realizar esta tarea. Básicamente, el software utilizado por el sistema consta de 2 grandes componentes: *(i)* MobileNetSSD para la detección, y *(ii)* un algoritmo de seguimiento y conteo automático basado en distancias de centroides (algoritmo de *tracking*).

En primer lugar, MobileNetSSD es un modelo detector de objetos que utiliza una arquitectura de red neuronal profunda para llevar a cabo las tareas de detección y clasificación de los vehículos(Howard et al., 2017; Liu et al., 2016). La clasificación realizada por el sistema se limi-

ta a reconocer cuatro clases de vehículos: automóviles, autobuses, motocicletas y bicicletas. En segundo lugar, se incorpora un algoritmo de seguimiento de vehículos basado en las distancias de los vehículos detectados por MobileNetSSD con el fin de realizar el conteo automático. Por otro lado, el hardware integrado al sistema consta elementalmente de un sensor llamado OAK-D y una computadora de placa reducida Raspeberry Pi. El OAK-D es el encargado de capturar los datos además de procesar la información correspondiente a la detección y clasificación de los vehículos. Por su parte, en la Raspberry Pi se ejecutará el algoritmo utilizado para el seguimiento y conteo automático.

En cuanto a los experimentos realizados, en primer lugar, se instaló un sistema integrado de detección, seguimiento y conteo de vehículos en la entrada principal del campus Chontalpa de la UJAT. Se establecieron distintas configuraciones de altura y de ángulo para el sensor OAK-D. Se empleó la red neuronal MobileNetSSD para la identificación y clasificación de vehículos según su tipo. Se aplicó un algoritmo de seguimiento basado en la distancia de centroides para contar los vehículos de manera automática. Los datos se recogieron en diferentes horarios y condiciones climáticas para asegurar la robustez del sistema bajo las diversas condiciones del ambiente. Además, se ajustaron parámetros determinantes como el umbral de detección y la distancia de seguimiento para optimizar la precisión del sistema.

Este sistema busca mejorar el control y agilizar la entrada de vehículos al campus Chontalpa de la Universidad Juárez Autónoma de Tabasco (UJAT). Otro de los objetivos del sistema es el de contribuir a la seguridad de los vehículos, del personal de la universidad, estudiantes, trabajadores y comerciantes, y de las instalaciones en general. Los resultados obtenidos en esta investigación confirman la viabilidad de un sistema automático de identificación y conteo de vehículos utilizando el detector MobileNetSSD, en combinación con un algoritmo de seguimiento basado en distancias de centroides. Este sistema logró una precisión y sensibilidad aceptables. La integración de tecnologías accesibles económicamente, como el sensor OAK-D y Raspberry Pi, no solo cumple con los objetivos específicos del proyecto, sino que también propone un sistema de bajo costo para ser replicado o mejorado.

1.2. Planteamiento del problema

1.2.1. Definición del problema

La gestión del acceso vehicular en el campus Chontalpa de la UJAT presenta desafíos significativos, especialmente debido a la cantidad de estudiantes y de personal distribuidos en las tres divisiones académicas que lo conforman. En la Figura 1.1 se muestra una toma aérea del campus Chontalpa de la UJAT. El marcador azul señala el acceso único al campus. Cada uno de los marcadores amarillos representa la ubicación de los 6 estacionamientos del campus. Las línea punteada negra/verde representan espacios en los cuales los vehículos pueden estacionarse en paralelo o en formato de batería. Finalmente, los marcadores rojos representan áreas que no son oficialmente para estacionarse, pero que debido al bajo número de plazas de estacionamiento disponibles son utilizadas como tal.

Figura 1.1. Ubicación de los espacios para estacionamiento del campus Chontalpa de la UJAT.

Con 234 plazas de estacionamiento y un elevado volumen de tráfico diario, es crucial contar con un sistema que administre tanto el tráfico como el estacionamiento. Esto es especialmente importante para evitar congestionamientos, especialmente durante las “horas pico”. Actualmente, el control de acceso se maneja de forma manual, lo que no solo consume tiempo y es propenso a errores, sino que también puede generar largas colas que interfieren con el tráfico adyacente.

Además, factores como la ubicación del sensor, el ángulo y la distancia entre la cámara y los vehículos, juegan un papel determinante en la calidad de los datos capturados. Por otro lado existen factores como la iluminación y el clima que no se pueden controlar de una manera práctica.

Estos factores pueden influir significativamente en los resultados obtenidos tanto por el detector como por el algoritmo de *tracking* y conteo automático basado en distancias de centroides.

A pesar de que existen diversos estudios y soluciones dirigidos a este problema, hasta el momento no se dispone de un modelo completamente confiable para la identificación de vehículos en tiempo real. Es fundamental, por lo tanto, continuar desarrollando y afinando estas tecnologías para mejorar su precisión y confiabilidad.

1.2.2. Delimitación de la investigación

- Esta investigación se concentra en el desarrollo y la implementación de un sistema automático de control de acceso vehicular para el campus Chontalpa de la UJAT.
- Este sistema integra el detector MobileNetSSD, el sensor OAK-D y una Raspberry Pi, con el fin de proponer un sistema simple y económico.
- El objetivo de este sistema es identificar y contar vehículos en tiempo real, además de generar informes estadísticos sobre el tráfico vehicular en el campus.
- Se utiliza el sensor OAK-D para la recolección de datos y el modelo MobileNetSSD para la detección y clasificación de vehículos.
- MobileNetSSD clasifica los vehículos en una de cuatro categorías: automóviles, motocicletas, autobuses o bicicletas.
- Se implementa un algoritmo de seguimiento basado en distancias para el conteo de vehículos.
- El sistema está diseñado para detectar únicamente los vehículos y no a los ocupantes de los mismos. Además, no es necesario guardar videos ni imágenes de los vehículos. Esto garantiza la protección de la privacidad de las personas.
- Esta investigación se enfoca geográficamente en el campus Chontalpa de la UJAT en el municipio de Cunduacán, Tabasco.
- El sistema se instala de forma fija en la entrada del campus, específicamente en la caseta de seguridad.

- Las pruebas se realizaron durante el horario de actividades del campus, aprovechando la luz solar de la mañana y omitiendo el uso de luz artificial.

1.3. Pregunta de investigación e hipótesis

Pregunta de investigación: ¿Cuál es el modelo de Aprendizaje profundo que proporciona una mayor precisión para identificar y contabilizar vehículos en tiempo real?

Hipótesis: Es posible crear un modelo basado en Aprendizaje profundo que identifique y lleve la cuenta de vehículos de manera automática con una precisión mayor al 80 %.

1.4. Objetivo general

Crear un sistema automático de detección y conteo de vehículos utilizando una arquitectura de Aprendizaje profundo y componentes de bajo costo.

1.5. Objetivos específicos

- Detectar los vehículos de acuerdo con sus características físicas mediante el uso del modelo MobileNetSSD.
- Realizar un conteo automático de vehículos mediante un algoritmo de *tracking* basado en distancias de centroides.
- Crear un reporte estadístico de los vehículos identificados.

1.6. Justificación

Azar et al. (2019) mencionan que la IA es utilizada en ámbitos en los que se registran tareas repetitivas, el uso de grandes volúmenes de información, riesgo de vida y extrema complejidad en la resolución de problemas. Esta tendencia es la motivación para enfrentar el problema de la identificación de vehículos mediante una técnica de IA. Más específicamente, de acuerdo a

lo mencionado por Albawi et al. (2017), el Aprendizaje profundo es considerado una de las herramientas más útiles y populares en la literatura gracias a su capacidad de manejar grandes volúmenes de datos. La implementación de una mayor cantidad de capas a las redes neuronales está empezando a superar el rendimiento de los métodos clásicos, especialmente en el campo del reconocimiento de patrones.

En cuanto al hardware, el sensor OAK-D sobresale como una herramienta viable y económica. Este sensor tiene la capacidad para realizar tareas de inteligencia artificial, como la detección y el seguimiento de objetos. Además, todo esto a un costo considerablemente bajo. Es importante tener en cuenta que una de las ventajas del Aprendizaje profundo sobre el enfoque de sensores es que no es necesario tener etiquetas para cada uno de los vehículos que se desea que se identifiquen.

Por su parte, la Raspberry Pi actúa como la columna vertebral computacional del sistema proporcionando un equilibrio entre rendimiento, portabilidad, economía y conexión. A pesar de su tamaño compacto y precio accesible, la Raspberry Pi es capaz de manejar las demandas de procesamiento del Aprendizaje profundo y de las tareas realizadas por el sensor OAK-D. Además, gracias a su conexión WiFi se facilita la comunicación con otras máquinas o servidores en la red. Esta combinación de hardware no solo demuestra que es posible implementar sistemas de IA con un presupuesto limitado. También favorece a la investigación y experimentación en el campo de la computación, donde los recursos pueden ser un factor limitante.

A grandes rasgos, este trabajo abarca los enfoques:

Computacional. Mediante la elección del Aprendizaje profundo, en particular redes neuronales convolucionales como enfoque para modelar el problema.

Social. Mediante el objetivo de procurar la seguridad y el orden entre los individuos y vehículos en el lugar del proyecto.

Económico. Mediante la aportación de un sistema simple que requiere muy poca tecnología y equipo.

1.7. Metodología utilizada

Para realizar un proyecto de ciencia de datos se debe realizar una serie de pasos imprescindibles. Estos pasos son: recopilar datos, limpiarlos, analizarlos, construir modelos y ponerlos en uso. Con el paso del tiempo se han propuesto diferentes marcos de trabajo con el fin generalizar el proceso de ciencia de datos de una manera formal, clara, detallada y organizada. Dos de los marcos de trabajo más conocidos son CRISP-DM (*Cross-Industry Standard Process for Data Mining*) y OSEMN (*Obtain, Scrub, Explore, Model, iNterpret*).

CRISP-DM es un estándar propuesto en 1996 por IBM que reduce costos y tiempo. Además, cumple con ser robusto, cuenta con técnicas independientes, apoya el entrenamiento y la transferencia de conocimiento, entre otros beneficios y objetivos. Peralta (2014) presenta una descripción detallada de cada una de las tareas que conforman la metodología CRISP-DM y que se pueden observar gráficamente en la Figura 1.2 que representa un diagrama del proceso de la metodología CRISP-DM.

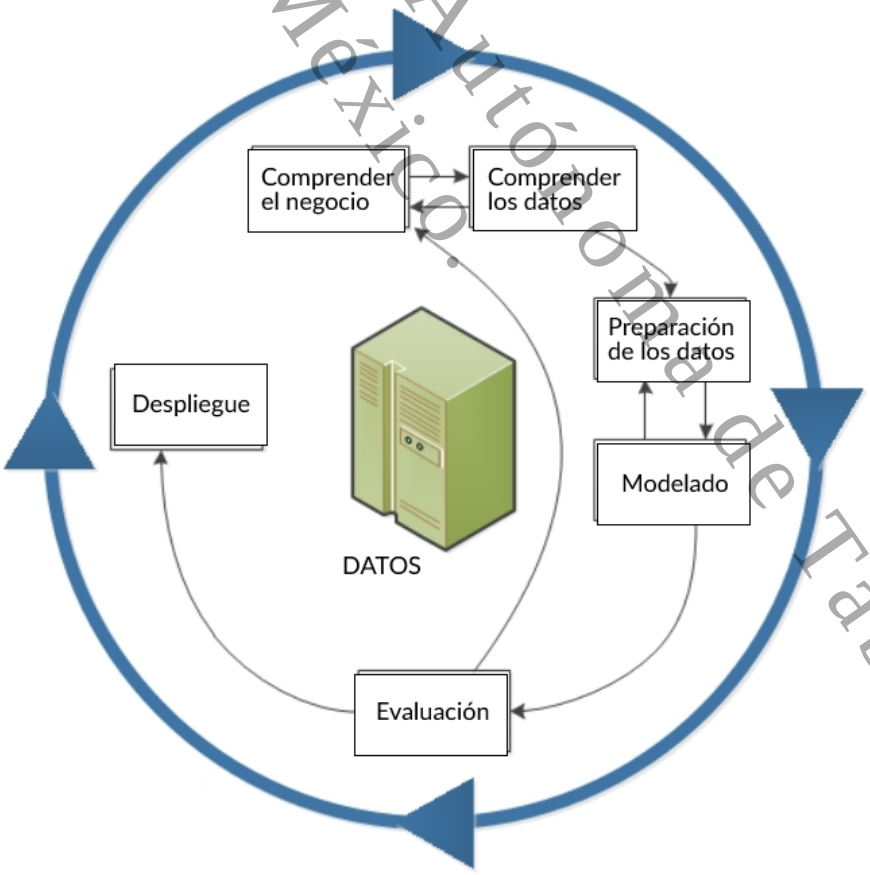


Figura 1.2. Diagrama de la metodología CRISP-DM (IBM-Corporation, 2021).

Por su parte, OSEMN es un marco de trabajo más reciente que fue propuesto y publicado por primera vez en 2010 por Mason & Wiggins (2010). El proceso OSEMN consta de cinco etapas: obtener, limpiar, explorar, modelar e interpretar los datos. Esta metodología o proceso puede observarse gráficamente en la Figura 1.3. OSEMN se centra en las fases o tareas a realizar y no en la organización que debe tener el equipo de trabajo (Saltz, 2019). Además, una de las diferencias claras entre CRISP–DM y OSEMN es que el segundo se salta las las fases iniciales de comprensión de los datos y del negocio de CRISP-DM así como el despliegue de los análisis resultantes (Lao, 2017).

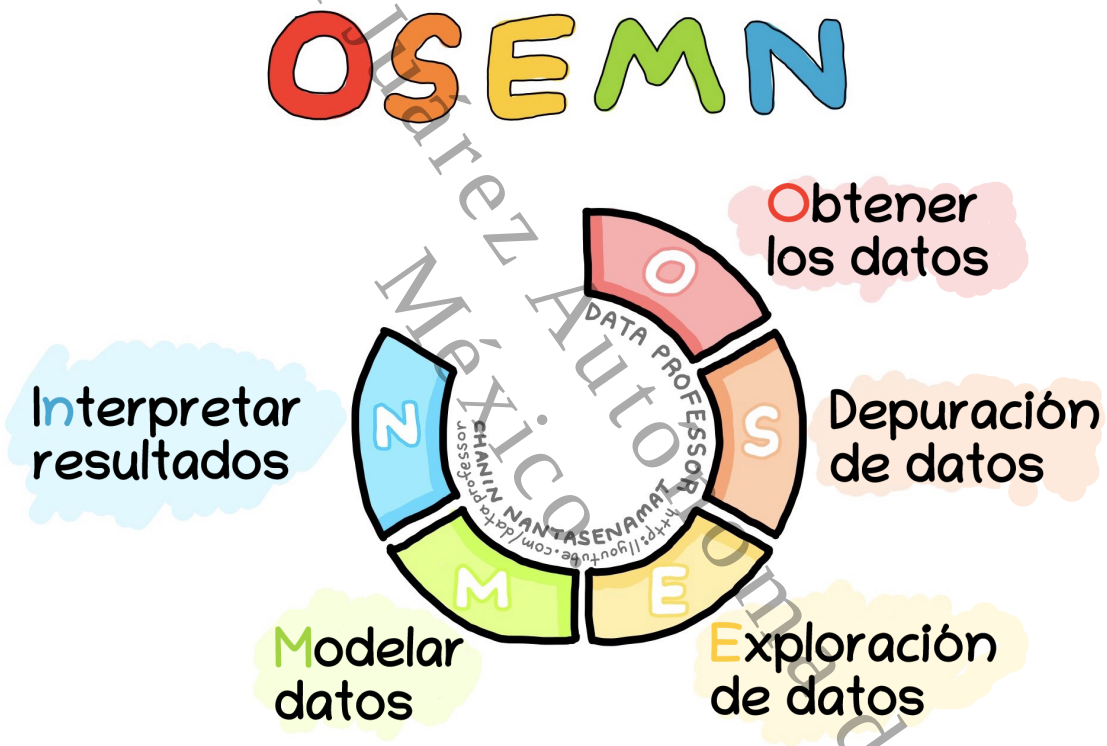


Figura 1.3. Diagrama de la metodología OSEMN (Lau, 2019).

Este trabajo se realizará con base en el marco OSEMN. A continuación se describen las etapas de OSEMN adaptadas a los objetivos de este proyecto.

Comprensión del negocio / Búsqueda de información. Primero, se realizó una investigación exhaustiva del estado del arte de la identificación de vehículos en tiempo real en diferentes aspectos. Se revisó cómo se ha abordado este problema mediante diferentes enfoques y objetivos. También se investigaron los principales lenguajes de programación, bibliotecas y

paquetes creados y utilizados en la literatura. Todo esto con el fin de utilizar el software adecuado para este fin. Dentro de este paso se identificaron tres grandes enfoques: el uso de sensores, el uso de algoritmos clásicos de la visión computacional y el Aprendizaje profundo. Por otro lado, se pudo reconocer a Python como el principal lenguaje utilizados dentro del estado del arte.

Obtener datos. Se realizaron experimentos en diferentes horarios de entrada de vehículos al campus con el fin de investigar aspectos como la ubicación y posición óptima del sensor. En este paso se establecieron los horarios, el tipo de iluminación, entre otros aspectos relacionados con la captura de los datos.

Limpiar datos. Posteriormente se dio tratamiento a los datos obtenidos con el fin de construir un conjunto de datos final. En este paso se busca eliminar información redundante o irrelevante así como darle el formato adecuado a los datos recopilados por el sensor. Debido a que las redes neuronales profundas pueden aprender directamente de los datos crudos, el preprocesamiento requerido fue mínimo. En este caso, solo fue necesario convertir las imágenes a la resolución de 300$\times$300 pixeles.

Modelado de los datos / Configuración de parámetros de la red.

Después de los primeros experimentos, se estableció el conjunto de prueba. Además, se probaron múltiples arquitecturas de red variando sus respectivos parámetros para encontrar el mejor modelo para este caso específico. En esta etapa se establecieron los valores de los parámetros del sistema. Los valores de *Threshold*, *Distance* y *Frames* fueron de 0.5, 500 y 20, respectivamente, para las categorías automóviles y autobuses. Por su parte, para las categorías motocicletas y bicicletas, los parámetros *Threshold*, *Distance* y *Frames* fueron de 0.5, 200 y 15, respectivamente.

Evaluación. Al finalizar la etapa de experimentación y la configuración de los parámetros, se pasó a la evaluación de la red. Esta prueba fue realizada con el sensor elegido (OAK-D), en la ubicación y posición descrita en el capítulo 4. Este proceso se realizó tomando en cuenta todos los puntos y variables anteriores. En esta fase también se evalúa y revisa los pasos ejecutados con relación a los objetivos propuestos en principio. En cuanto a los resultados,

si tiene que el modelo de detección obtuvo una *Precision* global de 0.81 y un *Recall* global de 0.91. Por su parte, el sistema integral de clasificación y conteo automático logró una *Precision* global de 0.80. Respecto a los objetivos planteados, el general y los específicos fueron alcanzados con éxito. Se desarrolló un sistema automático de detección y conteo de vehículos que emplea Aprendizaje profundo y componentes de bajo costo. Además se detectaron los vehículos con MobileNetSSD a partir de sus características físicas y se usó un algoritmo basado en distancias para realizar el seguimiento y conteo de los vehículos.

1.8. Descripción del documento

El presente documento está estructurado como sigue:

El Capítulo 1 aborda las generalidades del proyecto, exponiendo la introducción, planteamiento del problema, pregunta de investigación, hipótesis, objetivos generales y específicos, justificación, la metodología empleada y finalmente esta descripción del capitulado de la tesis. Este capítulo describe las bases para la comprensión del sistema propuesto.

En el Capítulo 2 se desarrollan los marcos conceptual, referencial, y tecnológico. Cubriendo así áreas cruciales como Aprendizaje automático y profundo, redes neuronales multicapa y convolucionales, detección y seguimiento de objetos. Además se expone una revisión de métodos existentes para la identificación automática de vehículos y se describe el software y hardware utilizados. Este capítulo contextualiza el proyecto dentro del amplio espectro de la inteligencia artificial en las ciencias de la computación.

El Capítulo 3 describe el sistema de detección, seguimiento y conteo de vehículos, incluyendo la integración de hardware y software, y el funcionamiento integral del sistema. Además, se justifica la elección de las tecnologías y métodos utilizados. Esta justificación se hace con base en experimentos y resultados preliminares.

El Capítulo 4 detalla los experimentos y resultados, desde la captura hasta el procesamiento de datos, y la ejecución práctica del sistema para la detección, clasificación y conteo automático de vehículos. Se discuten las pruebas realizadas, la forma de validación del sistema y la creación

de un reporte estadístico. Finalmente, se presentan los resultados detalladamente y las métricas de rendimiento.

Finalmente, el Capítulo 5 presentan las contribuciones, conclusiones y trabajos futuros. Con esto se resumen las aportaciones principales del proyecto y los resultados clave. Además que se sugieren líneas de investigación futuras.

Capítulo 2

Marco teórico

2.1. Marco conceptual

2.1.1. Aprendizaje automático

El Aprendizaje automático fue definido por Arthur Samuel como el estudio científico de los algoritmos y modelos estadísticos que los sistemas informáticos utilizan para realizar una tarea específica sin haber sido programados explícitamente para ello (Choudhary & Gianey, 2017). Los algoritmos de Aprendizaje automático son especialmente útiles en casos en los que implementar un algoritmo explícito con un rendimiento de alta velocidad es inviable. El principal objetivo del Aprendizaje automático es aprender de los datos (Mahesh, 2020). Es decir, en situaciones en la que un algoritmo explícito no es viable, nos apoyamos de los datos y se le dan instrucciones a la computadora para que les de un sentido inteligente a través de su análisis (Domingos, 2012). Estos algoritmos y modelos se utilizan para diversos fines, como la minería de datos, el procesamiento de imágenes, el análisis predictivo, entre otros. Es importante señalar que no existe un tipo específico de algoritmo que sirva para todo y que tenga mejores resultados que los demás. El tipo de algoritmo de Aprendizaje automático que se emplee dependerá del tipo de problema a resolver, el número de variables, el modelo que mejor se adapte a las condiciones reales, entre otros aspectos.

El nacimiento del Aprendizaje automático tuvo lugar en la década de 1950. En el año 1952 Christopher Strachey programa el primer algoritmo para jugar a las damas y posteriormente, en

1955 Arthur Samuel añade el aprendizaje a su algoritmo. Este fue el principal marco de Aprendizaje automático que obtuvo reconocimiento abierto. Con el paso de los años esta disciplina ha ido desarrollándose y creciendo constantemente. En el año 1967 se propuso el algoritmo del vecino más cercano (Alzubi et al., 2018). En 1981 Gerald Dejong propuso el *Explanation Based Learning* (EBL) con el cual una computadora puede analizar los datos de entrenamiento y crear reglas para descartar los datos inútiles (Minton & Zweben, 1993).

Por otro lado, en 1943 el neurofisiólogo Warren McCulloch y el lógico Walter Pitts trabajaban en el modelo McCulloch-Pitts de la neurona. Este modelo se basa en la clasificación de una observación en una de dos categorías (0 ó 1). La clasificación se realiza con base en un valor real correspondiente. La observación es clasificada como 1 si este valor supera un valor de umbral establecido y se clasifica como 0 si no supera el valor del umbral (Hayman, 1999). En adición a este modelo, en 1949 Donal Hebb propuso que cuando dos neuronas se activan juntas la conexión entre las neuronas se fortalece. Más adelante y con base en la neurona McCulloch-Pitts y los descubrimientos de Hebb, Frank Rosenblatt continuó con propuesta del perceptrón primario o regla de aprendizaje del perceptrón en el año 1957. Cinco años después en 1962, Rosenblatt demostró el teorema de convergencia del perceptrón (Shinde & Shah, 2018). Después de tres años de la demostración de este teorema, es propuesta por Bernard Widrowen la regla de aprendizaje Delta. Esta regla es utilizada para el entrenamiento del perceptrón y con esto se obtiene un buen clasificador lineal. Sin embargo, Marvin Minsky propuso en 1969 el problema XOR. Además demostró la incapacidad de los perceptrones en distribuciones de datos linealmente inseparables. A partir de esto, las investigaciones sobre las redes neuronales estuvieron inactivas hasta la década de 1980 (Chibuluma & Kalezhi, 2017).

Por otra parte, el Aprendizaje automático siguió creciendo durante la década de 1990 con diferentes técnicas y algoritmos. En 1995 Vladimir Vapnik y Corinna Cortes propusieron el mayor logro del Aprendizaje automático, las máquinas de vectores de soporte (SVM por las siglas en inglés de *Support Vector Machne*) (Cortes & Vapnik, 1995). A partir de esta fecha, el conjunto de investigadores de Aprendizaje automático fue aislado en dos grupos distintos: investigadores de SVM o investigadores de redes neuronales. En la siguiente sección se retoma el desarrollo de las redes neuronales artificiales. Por su parte, Yoav Freund y Robert E. Schapire desarrollaron en 1996 un conjunto reforzado de clasificadores débiles llamado Adaboost (Freund & Schapire,

1996). En 2001 Breiman explora Adaboost, ensamblando múltiples árboles de decisión donde cada uno de ellos es creado por un subconjunto aleatorio de instancias y cada nodo se selecciona de un subconjunto aleatorio de características. Este nuevo clasificador es conocido como Bosques aleatorios y demostró ser robusto al no tener los problemas de sobre-ajuste y de datos atípicos que presenta Adaboost (Shinde & Shah, 2018).

Los algoritmos de Aprendizaje automático se han aplicado para diferentes fines y en diferentes campos de estudio. Por ejemplo, en el campo de la Visión computacional se ha aplicado para el reconocimiento, la detección y el procesamiento de objetos, en el campo de la Predicción se ha aplicado para clasificación, análisis y recomendación. También se ha aplicado el Aprendizaje automático con fines de Análisis semántico, de Procesamiento del lenguaje natural y de Recuperación de información (Shinde & Shah, 2018).

En otro aspecto, el Aprendizaje automático se utiliza para resolver diversos problemas que requieren el aprendizaje por parte de una computadora y tiene tres características:

1. Task clases (la tarea a aprender).
2. Una medida de rendimiento que debe mejorarse.
3. El proceso de adquisición de experiencia

Además, un modelo de Aprendizaje automático consta de seis componentes, sin importar el algoritmo adoptado:

1. Recopilación y preparación de datos.
2. Selección de características.
3. Elección del algoritmo.
4. Selección de modelos y parámetros.
5. Entrenamiento.
6. Evaluación del desempeño.

Uno de los aspectos más interesantes de los algoritmos de Aprendizaje automático es el entrenamiento. Con base en la forma de entrenamiento y en la disponibilidad de la respuesta durante el entrenamiento, los algoritmos de Aprendizaje automático se pueden clasificar en 6 diferentes clases. Estas clases son: aprendizaje supervisado, aprendizaje semi-supervisado, aprendizaje no supervisado, aprendizaje por refuerzo, redes neuronales artificiales y aprendizaje basado en instancias (Alzubi et al., 2018; Laurent, 2022; Hao et al., 2016; Janiesch et al., 2021). Sin embargo, la clasificación más popular es: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo.

El aprendizaje supervisado consta de aprender una función que genere una respuesta a partir de un dato de entrada en particular con base en un conjunto de pares entrada-respuesta. Es decir, en el aprendizaje supervisado se cuenta con un conjunto de datos que incluyen las respuestas correctas y estos datos son usados para el entrenamiento del algoritmo. Una vez entrenado el algoritmo es capaz de generar respuestas para cada dato de entrada dado. El aprendizaje supervisado es utilizado comúnmente para regresión y clasificación. Algunos de los algoritmos de aprendizaje supervisado son: Árbol de decisión, Navïe Bayes, Máquinas de vectores de soporte, Análisis de regresión lineal, Regresión multivariada, Regresión logística, k vecinos más cercanos, entre otros (Choudhary & Gianey, 2017).

2.1.2. Aprendizaje profundo

El Aprendizaje profundo es una rama del Aprendizaje automático, cuya base principal es aprender características de una manera muy estratificada. La diferencia esencial entre el Aprendizaje profundo y otras técnicas es su capacidad de abstracción y de aprender jerarquías en diferentes niveles de representación. Las técnicas de Aprendizaje profundo buscan automatizar el proceso de abstracción, aprendiendo nuevas abstracciones a partir de las ya existentes (Berzal, 2019). En otras palabras, la diferencia entre el Aprendizaje automático y el Aprendizaje profundo es que en los algoritmos de Aprendizaje automático las características se extraen de forma manual y se obtiene un vector de características que será clasificado por el algoritmo de aprendizaje. Por su parte, en el Aprendizaje automático, las redes neuronales funcionan como extractor de características y también como clasificador (Elgendy, 2020). Por su parte, LeCun et al. (2015)

mencionan que el aspecto clave del Aprendizaje profundo es que las capas de características no están diseñadas por ingenieros humanos, sino son aprendidas a partir de los datos mediante un procedimiento de aprendizaje de propósito general.

El Aprendizaje profundo ofrece muchas ventajas gracias a su utilidad en diversos tipos de aplicaciones en el mundo real, una de las más importantes y poderosas es el poder trabajar con grandes volúmenes de datos (LeCun et al., 2015). Debido a esto, este tipo de técnicas han sido ampliamente estudiadas y son aplicables a una gran cantidad de problemas. Algunos de los dominios más explotados mediante el Aprendizaje profundo son: los negocios, la ciencia y el gobierno. Además de pruebas adaptativas, clasificación de imágenes biológicas, visión computacional, detección de cáncer, procesamiento de lenguaje natural, detección de objetos, reconocimiento facial, de escritura a mano, de voz, análisis bursátil y varios más (Dargan et al., 2020).

En general el Aprendizaje profundo ha demostrado un buen desempeño al enfrentar estos problemas. De hecho, en algunos de estos campos, se ha observado que al aumentar la cantidad de datos de entrenamiento, la exactitud de algunos algoritmos de Aprendizaje automático se ve superada por el Aprendizaje profundo.

La diferencia esencial entre el Aprendizaje profundo y otras técnicas es su capacidad de abstracción y de aprender jerarquías en diferentes niveles de representación. Las técnicas de Aprendizaje profundo buscan automatizar el proceso de abstracción, aprendiendo nuevas abstracciones a partir de las ya existentes.

Las técnicas de Aprendizaje profundo se basan en las redes neuronales artificiales. La principal diferencia entre las redes neuronales “tradicionales” y las redes neuronales profundas radica en la cantidad de capas y en las diferentes arquitecturas de redes utilizadas.

El aprendizaje supervisado es una técnica de aprendizaje que utiliza datos etiquetados. En los enfoques de Aprendizaje profundo supervisado, el entorno tiene un conjunto de entradas y salidas correspondientes y se les asigna un valor de pérdida. De manera iterativa se irán modificando los parámetros de la red de manera que en cada iteración se obtenga una mejor aproximación a los resultados deseados. En este enfoque se encuentran las redes neuronales profundas, las redes neuronales convolucionales, redes neuronales recurrentes, incluidas las de memoria a corto plazo y las unidades recurrentes con puertas.

2.1.3. Redes neuronales multicapa

Para poder comprender el concepto de Red neuronal artificial, primero vamos a hablar de su componente más básico, el perceptrón. El perceptrón es la red neuronal más simple y consiste de una sola neurona. Básicamente, el perceptrón está inspirado en las neuronas biológicas. Tanto las neuronas biológicas y las artificiales, reciben información proveniente de algunas neuronas, hacen algunos cálculos para ajustar la información recibida y crear otra información de salida que es enviada a otras neuronas.

En la Figura 2.1 se pueden observar los elementos de una neurona biológica y los de una artificial.

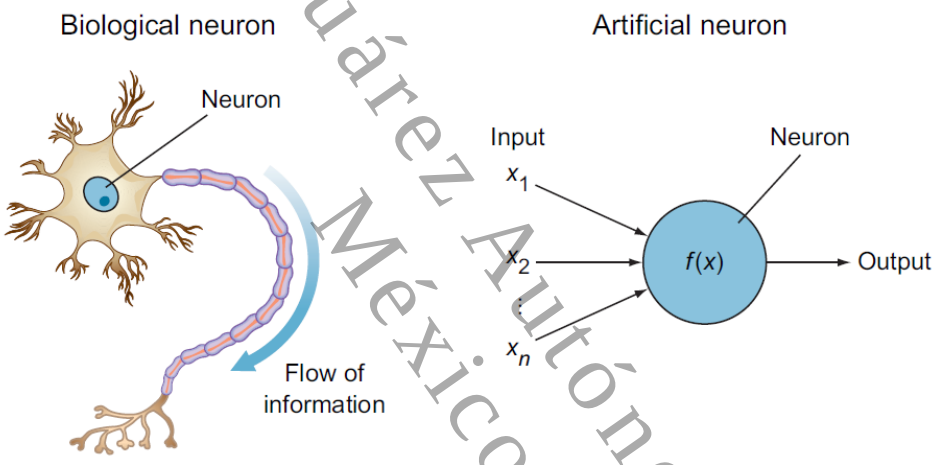


Figura 2.1. Las neuronas artificiales fueron inspiradas por las neuronas biológicas (Elgendy, 2020).

No todas las características de entrada son igual de importantes o útiles. Por lo tanto, cada una de estas características x_i tendrá asignado un peso w_i que represente su importancia en proceso de decisión, es decir, su efecto en el resultado. Entre mayor sea el peso de una característica, más se amplificará la señal de entrada de dicha característica y entre menor peso, se disminuirá dicha señal.

A continuación se mencionan los conceptos principales en el perceptrón y que pueden ser observados en la Figura 2.2:

- *Vector de entrada*. Es el vector X de características que alimenta a la neurona.
- *Vector de pesos*. Es el vector W de pesos que contiene los pesos correspondientes a cada

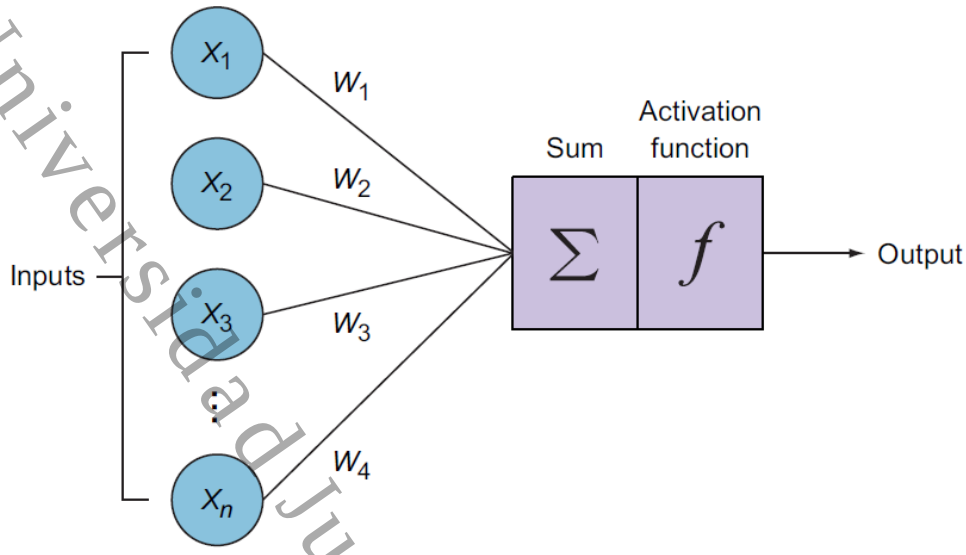


Figura 2.2. Diagrama de un perceptrón (Elgendy, 2020).

característica.

- *Funciones de la neurona*. Los cálculos realizados dentro de la neurona para ajustar las señales de entrada: la función suma ponderada y la función de activación.
- *Salida o resultado*. Es controlada por el tipo de función de activación elegido para la red. El nodo de salida o resultado representa la predicción del perceptrón.

El perceptrón utiliza prueba y error para aprender de sus errores. Los pesos son utilizados como perillas aumentando o disminuyendo sus valores hasta que el modelo es entrenado.

Como se mencionó anteriormente, el perceptrón es el modelo de red neuronal más simple y según Berzal (2019), fueron las primeras redes neuronales para las que se diseñó un algoritmo de aprendizaje en los años 50 del siglo XX.

Por otro lado, es natural preguntarnos si dicho modelo es suficiente para resolver problemas complejos. Tomando en cuenta que el perceptrón es una función lineal, es decir, la neurona entrenada genera como resultado una recta que separa o clasifica nuestros datos.

Como se puede intuir, existen problemas mucho más complejos y cuyos datos no pueden ser clasificados correctamente al separarlos por una sola recta. Debido a lo anterior, se introduce una clasificación para los problemas según el conjunto de datos. Dicha clasificación es definida

con base en dos conceptos simples que dieron lugar a los perceptrones multicapa: los problemas lineales y los no lineales.

Un problema lineal es un problema en el que el conjunto de datos puede ser separado o clasificado por una sola recta (Elgendy, 2020). y por el otro lado, un problema no lineal es en el que el conunto de datos no puede ser clasificado por una sola recta, es decir, se necesita más de una recta para generar una forma que separe los datos. En la Figura 2.3 se muestra un ejemplo del conjunto de datos de un problema lineal y el de uno no lineal. Además, en la Figura 2.4 se muestra un ejemplo de una pequeña red neuronal que es utilizada para modelar un conjunto de datos no lineales. En esta red neuronal se utilizan tres neuronas apiladas en una capa llamada *capa oculta*.

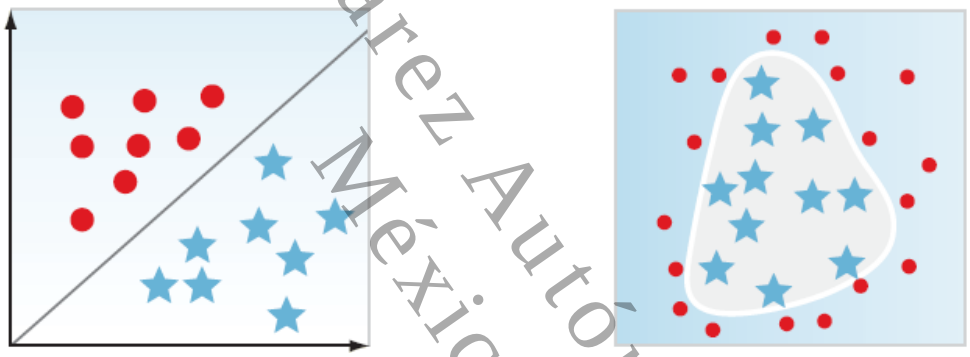


Figura 2.3. Ejemplos de conjuntos de datos lineales (izquierda) y no lineales (derecha) (Elgendy, 2020).

Como se puede observar en la Figura 2.4, una arquitectura de red neuronal es apilar las neuronas una encima de otra en capas ocultas. Cada capa tiene m número de neuronas y las capas están conectadas entre sí por conexiones de peso. Esta arquitectura es conocida como *perceptrón multicapa*. Los componentes principales de la arquitectura de red neuronal y que son descritos por Elgendy (2020) son:

- *Capa de entrada*. Contiene el vector de características.
- *Capas ocultas*. Las neuronas están apiladas una encima de otra. Estas capas son llamadas así debido a que no se puede ver ni controlar la entrada ni la salida de ellas. Sólo podemos alimentar la capa de entrada con las características y ver el resultado de la capa de salida.
- *Conexiones de peso*. Se asignan pesos a cada conexión entre neuronas con base en la

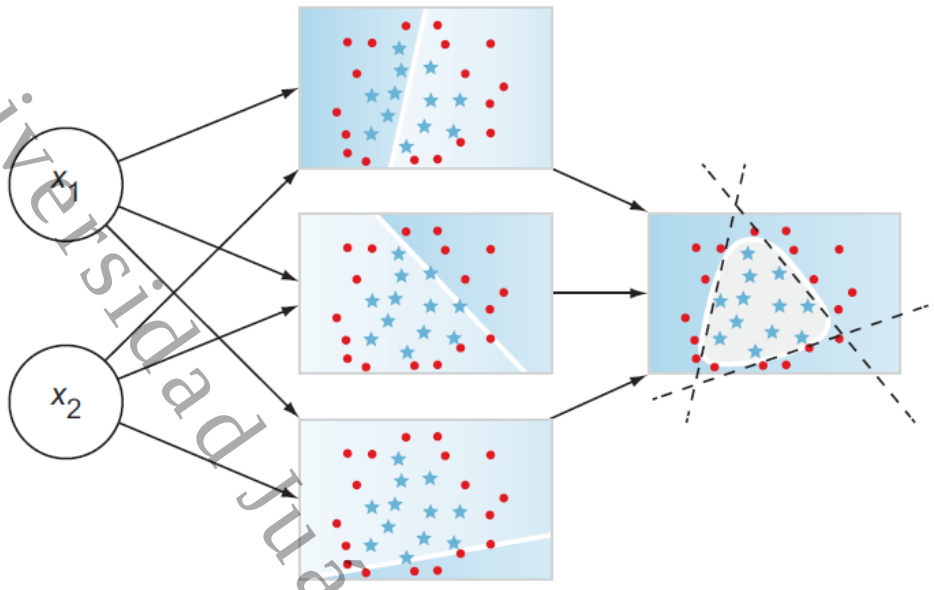


Figura 2.4. Diagrama de una red neuronal con 3 perceptrones en una capa oculta (Elgendy, 2020).

importancia de su influencia en la predicción final de la red.

- *Capa de salida*. De esta capa se obtiene la respuesta o predicción de nuestro modelo.

Es importante mencionar que en las capas ocultas es donde se encuentra la parte central del proceso de aprendizaje de características. Estas capas ocultas son apiladas para aprender características complejas unas de las otras. En la Figura 2.5 se ilustra un perceptrón multicapa (con seis capas ocultas) que busca modelar un conjunto de datos en espiral para distinguir entre dos clases. Note que las primeras capas ocultas detectan patrones simples para aprender características de bajo nivel (líneas rectas). Por su parte, las últimas capas ocultas detectan patrones dentro de patrones para aprender características y formas cada vez más complejas.

Otro punto muy importante dentro de las redes neuronales artificiales es el concepto de capa completamente conectada. Este concepto resulta ser muy intuitivo y simple, como su nombre lo indica, en este tipo de arquitecturas las capas están completamente conectadas con la siguiente capa oculta. Es decir, cada neurona en una capa está conectado con todas las neuronas de la capa anterior.

Los elementos y conceptos que se han desarrollado hasta ahora son parte de un tipo de arquitectura en específico. Se puede decir que esta arquitectura es la más básica de las redes

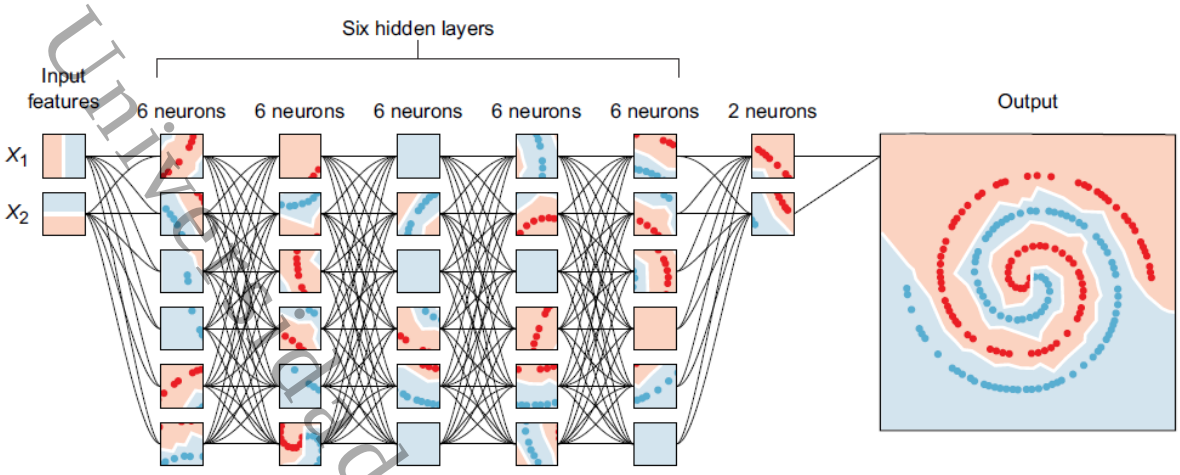


Figura 2.5. Diagrama de una red neuronal con 6 capas ocultas con 6 neuronas cada una (Elgendy, 2020).

neuronales artificiales y puede ser encontrada en diferentes textos como ANN (*Artificial Neural Network*), MLP (*Multilayer Perceptron*), *Fully connected network* o *Feedforward network*.

Como se observa en la Figura 2.2, cada neurona tiene un elemento llamado función de activación. Esta función es utilizada para convertir la entrada en una salida delimitada. Existen diferentes tipos de funciones de activación. El tipo de función de activación a utilizar varía según la situación. Sharma et al. (2017); Aloysius & Geetha (2017) mencionan que las principales funciones de activación son:

Sigmoide. La función sigmoide transforma los valores de entrada en un número entre 0 y 1. Los valores altos tienden de manera asintótica a 1 y de manera equivalente, los valores bajos tienden de manera asintótica a 0. La función sigmoide se define por la ecuación 2.1.

$$f(x) = \frac{1}{1 - e^{-x}}. \tag{2.1}$$

Tanh. La función tangente hiperbólica transforma los valores de entrada en un número en el intervalo $(-1, 1)$. Los valores altos tienden de manera asintótica a 1, mientras que los valores bajos tienden de manera asintótica a -1. La función tangente hiperbólica puede ser definida en términos de la función sigmoide como en la ecuación 2.2. Esta ecuación es equivalente a la expresada en la ecuación 2.3.

$$f(x) = 2sigmoide(2x) - 1. \tag{2.2}$$

$$f(x) = \frac{2}{1 - e^{-2x}} - 1. \tag{2.3}$$

ReLU. La función de unidad lineal rectificada transforma los valores de entrada negativos en 0, mientras que a los valores de entrada positivos los deja en su forma original. La función ReLU se define por la ecuación 2.4.

$$f(x) = \max(0, x) = \begin{cases} 0 \quad si \quad x < 0, \\ \\ x \quad si \quad x \geq 0. \end{cases} \tag{2.4}$$

Softmax. La función softmax transforma los valores de entrada en una representación de probabilidades. Es decir, los transforma en un número en el intervalo $(0, 1)$. Además, se debe cumplir que la sumatoria de todas las probabilidades de las salidas sea uno. La función softmax se define por la ecuación 2.5.

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}. \tag{2.5}$$

2.1.4. Redes neuronales convolucionales

Existen diferentes tipos de arquitectura de redes neuronales: redes neuronales multicapa, redes neuronales recurrentes, redes neuronales con unidades recurrentes con puertas, redes neuronales con memoria a corto y largo plazo y redes neuronales convolucionales.

Las redes neuronales convolucionales tienen su origen a partir de años de investigación para la mejora del perceptrón y las redes neuronales en general. Sin embargo, de manera más específica se podría decir que la arquitectura de las redes neuronales convolucionales nació a partir de la línea de investigación abierta en 1980 por el neocognitrón de Kunihiko Fukushima (Fukushima, 1980). El neocognitrón tenía la capacidad de reconocer patrones de estímulos basándose en la similitud geométrica de sus formas, sin verse afectado por sus posiciones. En la década siguiente, de las manos de Yann LeCun et al. nació LeNet, la arquitectura pionera de las redes neuronales convolucionales (LeCun et al., 1989, 1998). LeNet fue diseñada específicamente para clasificar

dígitos escritos a mano y resultó exitoso al reconocer patrones visuales directamente de la imagen de entrada sin procesar. Sin embargo, debido a la insuficiencia de datos de entrenamiento y de poder de cómputo, esta arquitectura no obtuvo buenos resultados en problemas complejos (Aloysius & Geetha, 2017). En 2012 Alex Krizhevsky et al. proponen la arquitectura AlexNet. AlexNet se volvió muy popular e influyente en el campo de la visión computacional desde el éxito obtenido en ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Krizhevsky et al., 2017). Además de el éxito en ILSVRC, AlexNet fue capaz de obtener mejores resultados que otros modelos existentes de redes neuronales convolucionales en diferentes trabajos.

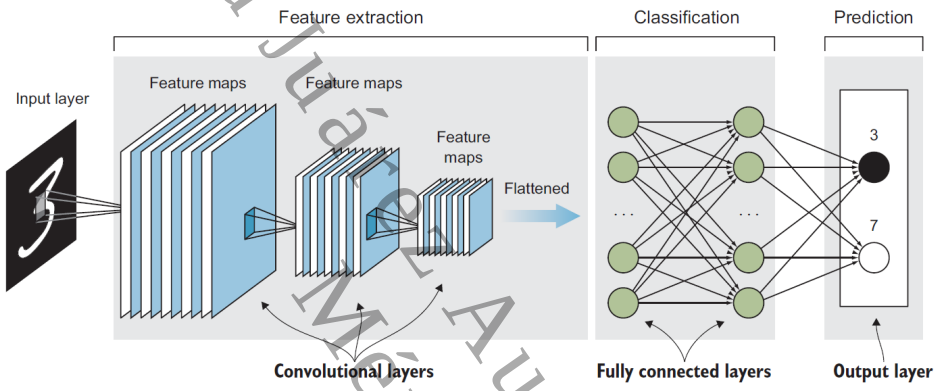


Figura 2.6. Diagrama de una red neuronal convolucional (Elgendy, 2020).

Las redes convolucionales o redes neuronales convolucionales son un tipo especializado de red neuronal artificial. Su principal característica es que alguna neurona de la capa oculta esta conectada solamente con un subconjunto de neuronas de la capa anterior. Debido a esta escasez de conectividad, la red es capaz de aprender características de forma implícita. Por otro lado, la arquitectura profunda de la red da como resultado una extracción de características de manera jerárquica. Es decir, las primeras capas de la red pueden detectar o reconocer patrones básicos, como los bordes y algunas manchas de color, mientras que las siguientes capas aprenden a reconocer formas básicas, como círculos, cuadros o ángulos, otras capas reconocen partes de objetos, como espejos, ojos, llantas, puertas, y finalmente, las últimas capas de la red son la que serán capaces de identificar los objetos como tal (Aloysius & Geetha, 2017).

El nombre de red convolucional proviene de la operación matemática convolución ya que ésta es realizada en la red. Con base en esto, Goodfellow et al. (2016) las definen como “simples redes neuronales que utilizan convolución en lugar de multiplicación matricial de entradas por pesos, en

al menos una de sus capas". La operación de convolución se aplica a una entrada mediante un filtro y resulta en una activación. Al realizar esta tarea repetidamente se obtiene un mapa de activación conocido como mapa de características. Las redes neuronales convolucionales son capaces de aprender diferentes filtros específicos y así obtener un conjunto de características que pueden detectarse en alguna parte de la imagen de entrada. En la Figura 2.6 se muestra el diagrama de una red neuronal convolucional y se pueden observar sus componentes principales: la capa de entrada, las capas convolucionales, las capas completamente conectadas y la capa de salida. Además, también se puede observar las tres tareas realizadas dentro de la red: extracción de características, clasificación y predicción.

Los tipos de capas que existen específicamente dentro de una red neuronal de tipo convolucional son las siguientes:

Capa convolucional. Es la unidad básica de una red neuronal convolucional y debe existir al menos una capa de este tipo. Hace referencia a la cantidad total de capas que utilizan la operación convolución sobre los datos de entrada. Por cada capa de este tipo se reduce la cantidad de características que se dan como entrada a la red densa. Sin embargo, por cada capa extra la mejora en la exactitud es cada vez menor. Por lo tanto, se debe encontrar un equilibrio entre la exactitud y la cantidad de capas convolucionales con el poder de cómputo y tiempo de entrenamiento. Los parámetros que controlan el volumen de la salida son la profundidad (números de filtros en una capa), *stride* (indica el tamaño del movimiento del filtro) y el *padding* (que está relacionado con explotar la información del "borde" de los datos).

Capa *pooling*. Las redes neuronales convolucionales tiene capas convolucionales y capas *pooling* alternadas. Las capas *pooling* reducen la dimensión espacial de los mapas de activación y el número de parámetros de la red. Con esto se consigue reducir la complejidad computacional general. Las operaciones *pooling* más comunes son *max pooling*, *average pooling*, y *stochastic pooling*.

Capa completamente conectada. Como su nombre lo indica, las neuronas de esta capa se encuentran totalmente conectadas a todas las neuronas de la capa anterior. Estas neuronas

no están espacialmente ordenadas y debido a esto no puede haber una capa convolucional despues de una capa completamente conectada.

Capa de pérdida. La última capa completamente conectada sirve para calcular la pérdida o el error. Esta pérdida o error es una penalización por la divergencia entre la salida real y la deseada. La función de pérdida más comúnmente utilizada es softmax loss. Esta función de pérdida es utilizada para predecir una única clase entre K clases mutuamente excluyentes. Para la regresión a valores reales se puede utilizar euclidean loss.

2.1.5. Detección de objetos

La detección de objetos es un área importante en la visión computacional y tiene numerosas aplicaciones prácticas, como sistemas de vigilancia (Elhoseny, 2020), navegación de vehículos autónomos (Hnewa & Radha, 2021) y análisis de imágenes médicas (Li et al., 2019). El objetivo de la detección de objetos es detectar todas las instancias de objetos de una o varias clases, como personas, vehículos o caras, en una imagen o secuencia de imágenes (Amit et al., 2020). Una parte esencial de este proceso es la representación de la ubicación y el tamaño de los objetos detectados, lo cual se logra a través de las *bounding boxes* o cajas delimitadoras. Un *bounding box* es un rectángulo que encierra el objeto de interés en la imagen, definiendo así su posición y sus dimensiones aproximadas. El objetivo de los algoritmos de detección de objetos es generar *bounding boxes* precisas que se ajusten a los objetos de la imagen, lo que permite su posterior análisis y procesamiento en sus diferentes aplicaciones (Redmon et al., 2016).

Históricamente se han propuesto varias técnicas y algoritmos para abordar el problema de la detección de objetos. En un principio, los enfoques eran basados en características manuales y técnicas de aprendizaje supervisado, como el clasificador de cascada Viola-Jones (Viola & Jones, 2001), entre otros, los cuales eran populares en la comunidad de investigación. Sin embargo, de la mano del Aprendizaje profundo y las redes neuronales convolucionales la detección de objetos ha experimentado un progreso significativo. Las redes neuronales convolucionales son especialmente efectivas para aprender representaciones jerárquicas de características visuales. Con esto, han demostrado un rendimiento superior en la detección de objetos en comparación con los enfoques tradicionales. Algunos de los *frameworks* y algoritmos más notables basados en

redes convolucionales para la detección de objetos incluyen R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015), SSD (Liu et al., 2016) y YOLO (Redmon et al., 2016).

Cada uno de estos enfoques tiene sus propias ventajas y desventajas en términos de precisión y velocidad de detección. Por ejemplo, YOLO es conocido por su velocidad de detección en tiempo real, pero a expensas de una menor precisión en comparación con Faster R-CNN y SSD. Por otro lado, Faster R-CNN y SSD ofrecen un equilibrio entre precisión y velocidad, lo que los hace adecuados para aplicaciones en tiempo real con requisitos de alta precisión.

2.1.6. MobileNetSSD

A partir de que AlexNet popularizó las redes neuronales profundas, la tendencia general ha sido hacer redes más profundas y complejas para mejorar la precisión. Sin embargo, en muchas aplicaciones del mundo real como la robótica, los coches autónomos y la realidad aumentada, las tareas de reconocimiento necesitan llevarse a cabo en tiempo real y en una plataforma computacionalmente limitada. Con base en este problema, surgió otra orientación para los modelos de redes neuronales que consiste en desarrollar redes más pequeñas pero eficientes. Sin embargo, muchos de estos trabajos sobre redes pequeñas se centran solo en el tamaño y no priorizan la velocidad. Este no es el caso de las MobileNets, que se centran principalmente en la optimización de la latencia, pero también produce redes pequeñas (Howard et al., 2017).

MobileNet es un modelo eficiente de red neuronal para aplicaciones de visión móvil y embebida. Se construye principalmente a partir de *depthwise separable convolutions* o convoluciones separables por profundidad para reducir el cálculo en las primeras capas. Por su parte, MobileNetSSD combina la eficiencia de las redes MobileNet con la precisión de *Single Shot MultiBox Detector* (SSD) (Liu et al., 2016). Esta combinación es capaz de realizar tareas de visión computacional que requieren reconocimiento y localización de objetos en tiempo real con restricciones computacionales.

MobileNet comienza con una única convolución estándar, seguida por 13 bloques de convoluciones separables en profundidad, cada uno compuesto por dos capas, una de convolución de profundidad y otra punto por punto (1×1), sumando un total de 27 capas de convolución. Cada capa incluye una normalización por lotes y una activación ReLU. Solo la última capa completamente conectada no utiliza ReLU y se enlaza directamente a una capa Softmax para clasificación. Las convoluciones 1×1 son usadas para combinar las características filtradas, reduciendo la complejidad computacional. Esta estructura facilita la personalización de la red mediante dos hiperparámetros: el multiplicador de anchura y el multiplicador de resolución. El primero ajusta la cantidad de canales por cada capa, y el segundo ajusta las dimensiones de entrada de los datos. Estos ajustes permiten modificar el modelo según las necesidades específicas de costo computacional y precisión del proyecto (Howard et al., 2017).

MobileNetSSD es efectivo para la identificación y conteo de vehículos gracias a su eficiencia

computacional y precisión. Incorpora SSD, que aumenta la capacidad de MobileNet para detectar y clasificar objetos en tiempo real. SSD también favorece y mejora la detección al utilizar *bounding boxes* que toman en cuenta diferentes tamaños y aspectos facilitando la captura de vehículos desde múltiples ángulos y distancias. Esto es beneficioso para aplicaciones de tráfico y de control de acceso vehicular.

2.1.7. Seguimiento de objetos

El seguimiento de objetos u *object tracking* es una tarea de la visión computacional en la que un algoritmo detecta objetos y sigue sus movimientos en el espacio a lo largo de una secuencia de imágenes o video. Es decir, un rastreador (o *tracker*) asigna una etiqueta única a cada uno de los objetos de interés detectados en los diferentes fotogramas de un video. Además, dependiendo del dominio de seguimiento, un rastreador también puede proporcionar información específica del objeto, como la orientación, el área o la forma de un objeto (Yilmaz et al., 2006). Existen muchos trabajos en lo que se han aplicado algoritmos de seguimiento a diferentes objetos y con distintos objetivos. Estas aplicaciones van desde seguimiento de humanos o personas con fines deportivos (Parmar & Tran Morris, 2017), con fines de salud pública (Singh Punn et al., 2020), pasando por el seguimiento de vehículos con fines de control de tráfico urbano (Tang et al., 2019) y de desarrollo de vehículos autónomos (Ravindran et al., 2021), hasta el seguimiento de peces con fines de conocer el comportamiento animal en la naturaleza (Spampinato et al., 2012) o de mejorar su explotación en granjas (Wageeh et al., 2021), entre muchas otras aplicaciones.

Si bien, el seguimiento de objetos está basado en la detección de objetos, en el primero, el objeto de interés se observa desde diferentes ángulos y distancias. Lo anterior provoca que dicho objeto pueda verse completamente diferente en algunos casos. Existen diferentes métodos para realizar la tarea del seguimiento de objetos. Algunos de los más conocidos e importantes son SORT (Bewley et al., 2016) y DeepSORT (Wojke et al., 2017). Estos algoritmos realizan la tarea de seguimiento de múltiples objetos en escenarios reales complicados. Además, son capaces de solucionar problemas como la oclusión, el *tracking* en tiempo real y las escalas espaciales variables en este tipo de escenarios. Sin embargo, existen algoritmos mucho más simples basados en la geometría del movimiento, las distancias y el tiempo.

Uno de los métodos funcionales y más simples es el descrito por Rahman et al. (2020). Este método es conocido como el algoritmo de seguimiento de objetos basado en centroides o algoritmo de distancias de centroides para el seguimiento de objetos. Para rastrear cada objeto, este algoritmo toma como entrada los cuadros delimitadores o *bounding boxes*. Estos cuadros delimitadores son generados por el detector (por ejemplo, MobileNetSSD). El centro de cada cuadro delimitador representa el centro de cada objeto. Cuando el centro del objeto entra en una Región de Interés (ROI, por las siglas Inglés de *Region Of Interest*), se le asigna un número de identificación único. En el fotograma siguiente los centros de los objetos se mueven o posiblemente se han quedado en el mismo lugar. El algoritmo de distancias de centroides para el seguimiento de objetos se basa en suposición de que cada objeto se moverá "muy poco" entre fotogramas subsequentes. De esta manera es posible relacionar cualquier centroide nuevo con un centroide antiguo que esté a una distancia mínima. Con este razonamiento podemos decir que un objeto ya fue previamente identificado y así actualizar la ubicación de su centro. Para realizar esta tarea, deben calcularse todas las distancias euclidianas posibles entre cada par de centroides antiguos y nuevos. La distancia euclidiana d entre dos pixeles (x_i, y_i) y (x_j, y_j) se calcula como sigue:

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2.6}$$

Si el fotograma actual tiene menos objetos que el fotograma anterior, significa que uno o más objetos han salido de la ROI. Debido a que sus centroides ya no están en la ROI, se elimina el número de identificación de la lista de objetos rastreados. Por otro lado, si el número de objetos en el fotograma actual es mayor que en el anterior, entonces debe haber un objeto nuevo. El algoritmo se actualizará y a los antiguos objetos se les asignarán nuevos centroides. Los objetos restantes serán tomados como objetos nuevos y se les asignará un nuevo número de identificación (Rahman et al., 2020).

2.1.8. Métricas de rendimiento

Existen diferentes métricas estándar que son la base de muchos estudios comparativos para determinar las mejores técnicas actuales de Aprendizaje profundo y de inteligencia artificial en general. Existen métricas de rendimiento (MOP) y métricas de eficacia (MOE) (Blasch et al., 2018).

En esta sección se expondrán las MOPs correspondientes al estado del arte de la detección de objetos.

Para hablar de las métricas de rendimiento de la detección de objetos debemos tomar en cuenta que muchos de los mejores algoritmos de detección han sido propuestos y puestos a prueba en competiciones mundialmente reconocidas. Algunas de estas competencias son VOC PASCAL Challenge (Everingham et al., 2010), COCO (Lin et al., 2014), ImageNet Object Detection Challenge (Russakovsky et al., 2015) y Google Open Images Challenge (Google AI, 2018).

Ahora se expondrán las métricas más utilizadas en el campo de la detección de objetos. Para esto, primero se enlistan los conceptos de verdaderos positivos, falsos positivos y falsos negativos. Además se hace mención del papel que tiene el concepto de verdadero negativo en la detección de objetos. Finalmente se mencionan los conceptos de las métricas *Precision* y *Recall* mostrando además su cálculo.

TP *(True Positives*). Detección correcta por parte del modelo (en realidad está el objeto y sí lo detecta).

FP *(False Positives*). Detección incorrecta por parte del modelo (en realidad no está el objeto y sí lo detecta).

FN *(False Negatives*). Detección no realizada por el modelo (en realidad sí está el objeto y no lo detecta).

Es importante mencionar que la cantidad TN (*True Negatives*) representa las detecciones no realizadas por el modelo cuando no está el objeto. En otras palabras, es un región correctamente no detectada por el modelo. TN no se utiliza en la detección de objetos porque existen muchos cuadros delimitadores en la imagen que no deberían detectarse (Padilla et al., 2020).

Dentro de la detección de objetos, la localización de diferentes objetos significa que la predicción genera un cuadro delimitador o *bounding box* alrededor de los objetos. La métrica utilizada para probar la exactitud de los cuadros delimitadores predichos es la IoU (*Intersection over Union*). La IoU indica el área de superposición del cuadro delimitador real (que localiza correctamente el objeto) BB_r y el cuadro delimitador predicho BB_p. En la ecuación 2.7 se muestra el cálculo del IoU entre dos cuadros delimitadores (Kaur & Singh, 2022).

$$IoU(BB_r, BB_p) = \frac{\text{área de intersección}}{\text{área de unión}} = \frac{\text{área}(BB_r \cap BB_p)}{\text{área}(BB_r \cup BB_p)}. \tag{2.7}$$

Ahora, para poder clasificar como correcta o incorrecta la predicción (detección de un objeto), además del IoU se debe tener un valor t conocido como umbral de confianza o *threshold*. Una detección es correcta si $IoU \geq t$; de otro modo, la detección se considera incorrecta. Además, si existen varias detecciones que se superponen con un mismo objeto real, sólo una se considera verdadero positivo (y las demás falsos positivos) (Cheng & Han, 2016).

Debido a la carencia de TN en la detección de objetos no se pueden utilizar las métricas tradicionales como *True Positive Rate* (TPR), *False Positive Rate* (FPR) y la curva ROC. Por esta razón, para la evaluación de los métodos de detección de objetos se basa en los conceptos de *Precision* y *Recall*. La *Precision* y el *Recall* se calculan como se muestra en las ecuaciones 2.8 y 2.9, respectivamente.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{todas las detecciones}}. \tag{2.8}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{todos los objetos reales}}. \tag{2.9}$$

La *Precision* es la capacidad de un modelo para identificar sólo los objetos significativos. Es decir, la proporción de predicciones positivas que son correctas. Por su parte, el *Recall* es la aptitud de un modelo para encontrar todos los casos significativos (objetos reales). En otras palabras, la proporción de predicciones positivas correctas entre todos los objetos reales. En términos generales, un detector debe identificar todos los objetos relevantes con el fin de encontrar los objetos reales. Para que un detector sea considerado un buen detector, el comportamiento de los valores de *Precision* y *Recall* deben mantenerse altos al hacer variaciones en el valor del umbral de confianza.

2.1.9. Detección automática de vehículos

El reconocimiento y la detección de vehículos han sido un foco principal de investigación en el campo de la visión por computadora y el Aprendizaje automático, dada su importancia en diversas

aplicaciones que incluyen la vigilancia y el monitoreo de tráfico (Wu et al., 2017).

Originalmente, se usaban predominantemente técnicas de visión por computadora para la detección de vehículos. Estos métodos, que a menudo dependen de características extraídas manualmente y técnicas de clasificación como máquinas de vectores de soporte, pueden ser sensibles a variaciones en la iluminación, las posturas de los vehículos y los cambios en el fondo (Zhang et al., 2016).

En contraste, las técnicas de Aprendizaje profundo, particularmente las redes neuronales convolucionales (CNNs), han mostrado un rendimiento superior en la tarea de detección de vehículos (Chen et al., 2014). Modelos como YOLO y SSD se han popularizado debido a su equilibrio de precisión y velocidad.

Liu et al. (2016) presentaron el SSD, una técnica que detecta objetos en imágenes utilizando una sola red neuronal profunda. Se mostró que SSD superaba a otros métodos de detección de objetos tanto en precisión como en velocidad.

De manera similar, Redmon et al. (2016) proponen YOLO, un enfoque de detección de objetos en tiempo real que considera la imagen completa durante la prueba, lo que le permite ser extremadamente rápido, manteniendo una precisión comparable a otros métodos de detección de objetos.

En este proyecto utilizamos el modelo MobileNetSSD, una variante de SSD diseñada para ser eficiente en términos de cálculo, lo que la hace adecuada para aplicaciones en tiempo real como la nuestra (Sandler et al., 2018).

2.2. Marco referencial

Existen muchas formas de identificar vehículos de manera automática, ya sea empleando técnicas de inteligencia artificial o técnicas de otras áreas de las Ciencias de la Computación. En esta sección mencionaremos algunos trabajos relacionados y con enfoques similares al nuestro.

Dentro del estado del arte de la Identificación automática de vehículos, se han utilizado principalmente tres enfoques: el uso de sensores, el uso de algoritmos clásicos del campo de la Visión computacional y el uso de Aprendizaje profundo.

2.2.1. Sensores

Griese & Kleinschmidt (2019) proponen un sistema para la identificación de vehículos usando RFID (*Radio Frecuency Identification*) y LPWAN (*Low Power Wide Area Networks*). Las matrículas de los vehículos son identificadas con etiquetas RFID y esta información es enviada a un servidor en la nube utilizando tecnología LoRa (*Long Range*). Posteriormente, se analizó el desempeño del sistema en entornos urbanos reales y resultó ser satisfactorio, tomando en cuenta diferentes criterios, principalmente la tasa de error. Las principales desventajas de esta propuesta es la adquisición de dispositivos e integración de diferentes tecnologías.

Por su parte, Álvarez Bazo et al. (2020) abordan el problema de la observabilidad en redes de tráfico. Los objetivos de este trabajo son: proponer una arquitectura para implementar redes de sensores de bajo costo capaces de ser instaladas temporalmente en la red de tráfico, un diseño de bajo costo y baja energía para el propio sensor, y un modelo para la ubicación del sensor capaz de establecer el mejor conjunto de enlaces de una red dados los objetivos de la investigación y las necesidades de instalación del sensor. Esto podría resultar un problema, en especial cuando el presupuesto es escaso y se busca minimizar los costos. Además, el tiempo de vida útil de un sensor se ve afectado por diversos factores como sobrecalentamiento, humedad, desperfecto por vibraciones, entre otros.

2.2.2. Visión computacional

Entre las publicaciones que abordan el tema mediante técnicas clásicas de Visión computacional se encuentra la de Yang et al. (2016), en donde muestran que es posible identificar un vehículo registrado usando un solo ejemplo de entrenamiento. El punto clave es describir propiedades representativas del vehículo, como la apariencia general y la textura local mediante colores e histogramas de gradientes orientados (HOG). Este enfoque fue puesto a prueba por medio del conjunto de datos abierto “Dana36” utilizando imágenes de vehículos que son tomadas bajo una posición de cámara aproximada.

Por su parte, Vibha et al. (2018) presentan un *framework* cuyo objetivo es monitorear las actividades en las intersecciones de tráfico para detectar congestiones y luego predecir el flujo de tráfico, lo que ayuda para regular el tráfico. Se utiliza un algoritmo basado en Visión compu-

tacional para la detección y recuento de vehículos en secuencias de imágenes monoculares para lugares de tráfico. Para obtener dichas imágenes se utiliza una cámara fija. El método se basa en establecer correspondencias o relaciones entre regiones y vehículos conforme se mueven por la secuencia de imágenes. Se utiliza la sustracción de fondo, que mejora el modelo de mezcla de fondo adaptativo y a su vez ayuda a que el sistema aprenda de manera más rápida y más precisa, además de adaptarse de forma eficaz a entornos cambiantes. Finalmente, el sistema resultante identifica los vehículos en las intersecciones, elimina el fondo y rastrea los vehículos durante un periodo específico de tiempo.

Ahmad & Boufama (2019) también utilizan HOG. Sin embargo, a diferencia de Yang et al. (2016), en su trabajo toman en cuenta un detalle importante. En su publicación proponen un sistema de identificación de vehículos que no tome en cuenta solo la forma del vehículo, sino también su matrícula. Primero se detecta la placa del vehículo, después, se reconoce el texto de la matrícula seguido de su provincia de origen (de la matrícula), y finalmente, se detecta la forma del vehículo. En el sistema, las características son convertidas en un patrón binario local (LBP) y un histograma de gradientes orientados (HOG) como dataset de entrenamiento. Además, fue utilizado un método basado en clasificadores en cascada para actualizar el sistema. Finalmente, se guardaron las características de los vehículos en la base de datos y de este modo se hizo posible detectar automáticamente discrepancias entre una matrícula y el vehículo asociado a ella.

2.2.3. Aprendizaje profundo

En este paradigma de la identificación de vehículos, generalmente se utilizan sensores para reconocer vehículos en tiempo real, debido a la mayor precisión obtenida por el previo etiquetado de los vehículos y esto garantiza de alguna manera el reconocimiento automático. Sin embargo, en la últimas fechas el Aprendizaje profundo está teniendo gran auge en este campo debido a que no se requiere la compra de dispositivos adicionales.

Mondal et al. (2017) abordan el problema del robo de vehículos y la vigilancia de las reglas de tránsito mediante el procesamiento de imágenes. Con base en las capacidades de aprendizaje de las redes neuronales convolucionales (CNN), se busca detectar y reconocer el número de ma-

trícula del vehículo. Las imágenes de las matrículas son obtenidas mediante una cámara fija. La característica auto sintetizada de la red neuronal convolucional es capaz de reconocer el vehículo a partir de la matrícula con una precisión del 90 %. La red neuronal convolucional ha demostrado ser robusta incluso utilizando conjuntos de datos distorsionados, inclinados e iluminados.

Selmi et al. (2017) también abordan el problema de la detección y reconocimiento de la matrícula de un vehículo. Dicho problema ha demostrado ser muy complejo debido a diferentes detalles, por ejemplo, las características de la placa que varían según el estado o país de origen (colores, tamaño, tipo de letra, etc.), la calidad de las imágenes de los vehículos, la velocidad de movimiento de los vehículos, iluminación, condiciones climáticas, entre otros factores. Debido a estas limitantes, en este trabajo se presenta un sistema automático de detección y reconocimiento de matrículas basado en el Aprendizaje profundo. Dicho sistema está dividido en tres partes: detección, segmentación y reconocimiento de caracteres. El rendimiento del sistema se prueba en dos conjuntos de datos que contienen imágenes en diversas condiciones como mala calidad de imagen, distorción de la perspectiva de la imagen, día soleado o noche y en entornos complejos. Un alto porcentaje de aciertos mostraron la precisión del sistema.

Yang et al. (2018) proponen un método de detección automática de vehículos basado en el *framework* YOLO versión 2. El método mejora a YOLOv2, optimizando los parámetros del modelo. Esto logra que el modelo pueda aprender automáticamente las características del vehículo y realizar una detección automática de vehículos en tiempo real y con una precisión mayor en comparación con YOLOv2.

Izidio & Barros (2018) proponen una solución para detectar y reconocer matrículas de autos mediante redes neuronales convolucionales. El sistema se implementó en una Raspberry Pi 3 con un módulo de cámara Pi NoIR v2 para obtener las imágenes de los vehículos. El sistema detecta las placas en la imagen capturada utilizando la arquitectura Tiny YOLOv3 e identifica sus caracteres utilizando una segunda red neuronal convolucional entrenada sobre imágenes sintéticas y afinadas con imágenes de placas reales. El sistema demostró ser lo suficientemente robusto para variaciones en ángulos, iluminación y ruido. El sistema se validó con base en imágenes de placas reales bajo diferentes condiciones ambientales alcanzando una tasa de detección del 99.37 % y una tasa de reconocimiento general del 97.00 % mientras mostraba un tiempo promedio de 2.70 segundos para procesar imágenes de tamaño 1024×768 con una sola matrícula.

Chen (2019) también aborda el problema del reconocimiento automático de matrículas de vehículos pero utilizando un marco de Aprendizaje profundo de Darknet YOLO. Se utilizan sus 7 capas convolucionales para detectar solo una clase. El método de detección es un proceso de ventana deslizante. El objetivo es reconocer las matrículas de los automóviles de Taiwán. La ventana detecta cada uno de los 6 dígitos de la matrícula y luego cada ventana es dectectada por un solo marco YOLO. El sistema logra un 98.22 % de precisión en la detección de placas y un 78 % en el reconocimiento de las matrículas. El sistema ejecuta una única fase de reconocimiento de detección, que necesita alrededor de 800ms a 1s para cada imagen de entrada. Este sistema también fue probado en condiciones complejas como un fondo lluvioso, en oscuridad y penumbra, y con diferentes tonos y saturación de imágenes.

Tourani et al. (2020) proponen un método para la detección de placas de vehículos iraníes y el reconocimiento de caracteres como una aplicación conjunta que presenta precisión significativa y un rendimiento en tiempo real. Este método funciona en placas iraníes con diferente resolución y diseño así como dígitos y caracteres escasos. El sistema propuesto utiliza una plataforma de la versión 3 de YOLO ajustada por separado para cada una de las fases mencionadas y extrae caracteres persas de las imágenes de entrada en dos pasos automáticos. Los resultados experimentales muestran una precisión del 95.05 % en 5719 imágenes. Los datos de prueba incluyeron imágenes en color y escala de grises que contenían los vehículos con diferentes distancias y ángulos de disparo con varios brillos y resoluciones. Además, el sistema puede realizar las tareas LPD y CR en un promedio de 119.73 milisegundos para datos de la vida real, lo que ilustra un rendimiento en tiempo real del sistema y una aplicabilidad utilizable. El sistema es completamente automático y no necesita de preprocesamiento, calibración o configuración.

Por su parte, Tang et al. (2021) abordan el problema de la estimación dinámica del flujo *Origin-Destination* (OD). Se propone una red neuronal profunda basada en convolución tridimensional, “Res3D” para aprender las correlaciones de alta dimensión entre los patrones de tráfico local presentados por las observaciones de identificación automática de vehículos y los flujos OD. Se presenta también un *framework* práctico que combina el entrenamiento de modelos basado en simulación y el aprendizaje de transferencia *few-shot* para mejorar la aplicabilidad del modelo propuesto, debido a que la observación continua de los flujos OD podría ser muy costosa. El modelo fue probado en una red de carreteras realista y los resultados muestran que para flujos

OD significativos, los errores relativos son estables alrededor del 5 %, superando a otros modelos existentes.

Un trabajo muy importante dentro del estado del arte es el de Wang et al. (2019). En él se realiza una revisión de las principales arquitecturas de Aprendizaje profundo utilizadas en la detección de vehículos, destacando principalmente RetinaNet, gracias a su precisión de detección bastante alta, que a su vez, es consecuencia de una función de pérdida que puede reducir efectivamente el peso de las muestras fáciles de clasificar y así centrarse en las muestras difíciles en la fase de entrenamiento. Además, por medio de experimentos comparativos, se encuentra que SSD (*Single Shot MultiBox Detector*), a pesar de ser antiguo, presenta una gran ventaja en tiempo real en comparación a YOLOv3. Por otro lado, en las pruebas realizadas en escenarios reales, se descubre que SSD también posee una excelente capacidad de generalización ya que la construcción del modelo es menos propensa al sobreajuste y cumple ser robusto.

2.3. Marco tecnológico

Para el desarrollo de nuestra propuesta se requirió el siguiente equipamiento de hardware y software:

Sensor OAK-D. El dispositivo OAK[1] (OpenCV *Artificial Intelligence Kit with Depth*) cuenta con tres cámaras integradas que implementan la visión estereoscópica y RGB, que a su vez están acopladas a un procesador de la marca Intel llamado Myriad X inside VPU capaz de ejecutar modelos de IA en un formato específico.

Mobilenet SSDv2. Este modelo de red neuronal convolucional, al igual que la mayoría de los modelos de redes ligeras utilizan a Mobilenet-v2 como red troncal (Howard et al., 2017; Liu et al., 2016). Esta incluye una capa convolucional estándar y 17 módulos residuales inversos. Por su parte, cada módulo residual inverso contiene una capa convolucional 1×1, una capa convolucional separable Dwise 3×3, funciones de normalización por lotes (*batch normalization*) y función de activación ReLU.

[1] `https://docs.luxonis.com/projects/hardware/en/latest/pages/BW1098OAK/`

DepthAI. DepthAI[2] es una plataforma de alto rendimiento de visión computacional e inteligencia artificial espacial que permite a dispositivos como robots y computadoras, a comprender y actuar en su entorno físico[3]. En general, se centra en la inteligencia artifical, en la visión computacional, en la percepción de la profundidad mediante dos cámaras (estéreo) o mediante tecnología *Time of Flight* (ToF), en el rendimiento mediante múltiples sensores, una alta resolución y los *frames* por segundo (FPS) y en ser una solución embebida de bajo consumo facilitando su integración a diferentes tipos de sistemas.

Raspberry Pi 4 modelo B. Raspberry Pi es un ordenador de placa reducida o placa única (SBC – *SingleBoard Computer*) de bajo coste (Aldea, 2017). Este SBC es desarrollado en Reino Unido por la Fundación Raspberry Pi. Raspberry Pi 4 B utiliza un procesador quad core cortex a 1.8 GHz, memoria RAM (2, 4 u 8GB) y tarjeta gráfica o GPU (*Graphics Processing Unit*, Unidad de Procesamiento Gráfico) en un solo chip, por tanto se trata de un sistema SoC (*System on a Chip*, Sistema en un chip). El soporte para conexión WiFi IEEE 802.11ac y la incorporación de dos puertos USB 3.0 son características destacadas de la Raspberry Pi 4 B, ya que proporcionan una conectividad esencial para el funcionamiento de dispositivos periféricos, como el sensor OAK-D.

[2] https://docs.luxonis.com/en/latest/

[3] https://docs.luxonis.com/en/latest/pages/faq/#what-is-depthai

Capítulo 3

Sistema de detección, seguimiento y conteo de vehículos

En este capítulo se describe en detalle el sistema de detección, seguimiento y conteo automático de vehículos en el campus Chontalpa de la UJAT. Además, se mencionan algunas pruebas de experimentos preliminares que están relacionadas con la elección del hardware y el software, la configuración y posición del sensor, la clasificación de los vehículos, entre otros detalles.

3.1. Diagrama del sistema

El sistema desarrollado integra hardware y software especializado en visión computacional y redes neuronales convolucionales para lograr la detección, clasificación, seguimiento y conteo automático de los vehículos que ingresan al campus Chontalpa de la UJAT. En esta sección se describe cómo los componentes trabajan conjuntamente y las razones detrás de la selección de cada componente.

En la Figura 3.1 se muestra el diagrama integral del sistema y se describen de manera breve cada una de las tareas realizadas. En específico, dentro del sensor OAK-D se ejecuta la red neuronal MobileNetSSD para la detección y clasificación de los vehículos. Por su parte, en la Raspberry Pi 4 se ejecuta el algoritmo de tracking basado en distancias de centroides que es complementado con un algoritmo que realiza el conteo de vehículos de manera automática.

A continuación, se mencionan algunos resultados preliminares que nos indican como la posi-

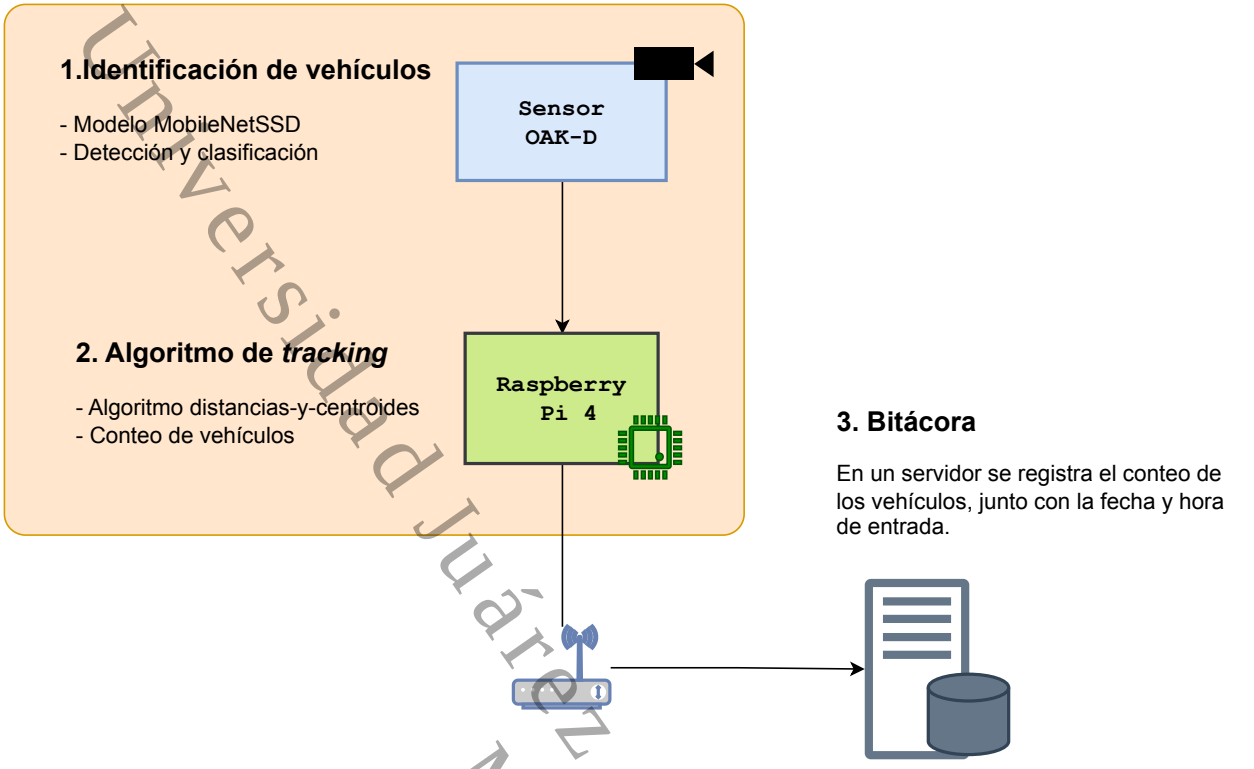


Figura 3.1. Diagrama integral del sistema.

ción, el ángulo y la altura del sensor impactan en la calidad de la detección de vehículos. Estos resultados preliminares fueron importantes al momento de integrar el hardware y el software. Estas configuraciones tienen el objetivo de maximizar la precisión de la detección y del seguimiento (*tracking*) de vehículos.

3.2. Integración del hardware

Sensor OAK-D y Raspberry Pi 4. Para la recolección de datos se utilizó una cámara OAK-D con base en su capacidad para realizar tareas de Aprendizaje profundo ya que cuenta con una unidad de procesamiento de visión artificial o VPU (por las siglas en Inglés de *Vision Processing Unit*) integrada. Una VPU es similar en potencia GPU pero especializada en procesamiento de imágenes y más eficiente en términos de consumo de energía.

El sensor OAK-D está conectado a la Raspberry Pi 4, la cual actúa como un puente entre el sensor y la red UJAT. La Raspberry Pi 4 proporciona suministro de energía al sensor, además de procesar parte de la información capturada por el sensor en tiempo real. No

obstante, el VPU Myriad X incorporado en el OAK-D permite que la mayor parte del procesamiento, incluida la detección, se realice dentro del propio sensor. Esto alivia la carga de procesamiento en la Raspberry Pi 4.

Por su parte, la Raspberry Pi 4, cuenta con capacidades de conectividad WiFi, lo que permite transmitir los datos obtenidos por el sistema a un servidor de la universidad mediante la red WiFi interna. Esta característica ayuda garantizar una transmisión de datos eficiente para la comunicación.

La elección de la Raspberry Pi 4 se basó en algunos factores como su bajo costo, su capacidad de procesamiento, sus dimensiones relativamente pequeñas, su capacidad de conexión WiFi y su facilidad de conexión e integración con el sensor OAK-D. En cuanto a la elección del OAK-D, este es reconocido actualmente por sus aplicaciones de visión computacional e inteligencia artificial. Además, existe una gran cantidad de algoritmos y modelos ya entrenados y optimizados para su uso con la OAK-D. Estos modelos van desde reconocimiento facial o de peatones hasta detección y seguimiento de objetos incluidos vehículos. En la Figura 3.2 se observa el sensor OAK-D y la Raspberry Pi 4 integrados. La forma de conexión es mediante un cable USB 3.0 a USB tipo C.

Un aspecto crítico para mejorar el rendimiento del sistema fue la ubicación, el ángulo y la altura a la que se colocó el sensor OAK-D. Se realizaron experimentos preliminares para determinar las condiciones óptimas que permitirían obtener una toma clara y detallada de los vehículos que se acercaban. Cabe destacar que el sensor no se ubicó de frente a la vía, en su lugar, se colocó ya sea en el lado derecho o izquierdo del carril de entrada, ajustando su ángulo y altura para optimizar la calidad de la captura. Estos experimentos consistieron en variar los rangos de altura y ángulo del sensor para evaluar la calidad y precisión de las detecciones realizadas por el modelo. Es importante mencionar que no existen topes o flechas en la entrada para reducir la velocidad de los vehículos, esto influye directamente en el comportamiento de los vehículos que entran. Los resultados de estos experimentos mostraron que a una altura de entre 1 y 1.8 metros del suelo y a un ángulo de entre -15 y 15 grados respecto a la línea de entrada al campus se maximizaba la precisión en la obtención de imágenes de calidad para la detección y el seguimiento de los vehículos.

Figura 3.2. Sensor OAK-D y Raspberry Pi 4 integrados.

Además, es importante resaltar que la elección del sensor OAK-D no solo se basó en sus capacidades técnicas, sino también en su accesibilidad económica y en las bondades de características que brinda, especialmente gracias a su VPU integrada. Debido a que el sensor OAK-D es un dispositivo relativamente nuevo y accesible, su estudio y aplicación en proyectos de visión computacional, inteligencia artificial y de Aprendizaje profundo, son de gran interés para la comunidad científica (Rojas-Perez & Martinez-Carranza, 2021; Perazzo et al., 2022; George & Marcel, 2020, 2021; Saoudi et al., 2023). Su capacidad de procesamiento en tiempo real es ideal para tareas complejas como la detección y seguimiento de vehículos gracias a la VPU integrada.

Al combinar la conectividad y portabilidad de la Raspberry Pi 4 con la capacidad del sensor OAK-D para implementar modelos de Aprendizaje profundo y redes neuronales convolucionales, se crea un sistema visión computacional con fines de control de acceso vehicular capaz de detectar, clasificar, rastrear y contar los vehículos entrantes automáticamente.

En la Figura 3.3 se muestran las 8 ubicaciones en las que se posicionó el sensor (círculos amarillos). Estas 8 ubicaciones incluyen las utilizadas en experimentos preliminares y las utilizadas en el experimento principal. Las ubicaciones definitivas están marcadas mediante los números 1, 2 y 3. Finalmente, el área en color azul representa la ROI (*Region Of Interest*). La delimitación de la ROI es crítica, en especial al calcular las métricas de rendimiento expuestas en las Ecuaciones 2.8 y 2.9.

3.3. Integración del software

DepthAI, MobileNetSSD y Algoritmo de tracking. El software principal del sistema fue desarrollado en lenguaje Python, específicamente la versión 3.9.7. Se utilizó la biblioteca DepthAI en su versión 2.23.0.0[1] para interactuar con el sensor OAK-D. Además, se hizo uso del sistema operativo Raspberry Pi OS (64-bit), propio de la Raspberry Pi 4. Por su parte, MobileNetSSD en su versión 2 puede detectar y clasificar 21 clases diferentes de objetos en cada uno de los *frames* que reciba como entrada. Sin embargo, con base en la tarea a realizar,

[1] `https://pypi.org/project/depthai/#history`

Figura 3.3. Posiciones del sensor y ROI.

la clasificación de los vehículos realizada por MobileNetSSD fue limitada a cuatro clases:

1. Automóviles.
2. Motocicletas.
3. Autobuses.
4. Bicicletas.

Una vez detectados y clasificados, el algoritmo de seguimiento (*tracker*) rastrea los vehículos a lo largo de las secuencias de fotogramas. Este algoritmo está basado en las distancias

de los centroides de las detecciones realizadas por MobileNetSSD. La tarea de seguimiento o *tracking* de los vehículos nos permite realizar un conteo de forma automática.

DepthAI fue elegida debido a su compatibilidad y facilidad de integración con la cámara OAK-D y a su vez con MobileNetSSD. Por su parte, MobileNetSSD destacó por su equilibrada relación entre precisión y una casi perfecta sensibilidad (*Recall*) a distancias cortas. Las pruebas preliminares realizadas en el campus Chontalpa reflejaron la superioridad de MobileNetSSD. A diferencia de otros modelos, este demostró ser el más eficiente en la detección de vehículos en tiempo real, reafirmando que estos resultados son especialmente válidos cuando el modelo se implementa dentro del sensor OAK-D.

El algoritmo de seguimiento de distancias de centroides fue seleccionado principalmente con base en su simplicidad y debido a su uso para el conteo automático en otras propuestas donde se utiliza *tracking*. Este algoritmo, a pesar de ser muy simple, resulta eficaz para realizar la tarea de seguimiento y a su vez la del conteo automático. Aún así, a pesar de ser una opción viable para realizar las tareas de interés, deben calibrarse o ajustarse algunos parámetros para optimizar los resultados del conteo automático. Algunos de estos parámetros son el ángulo y la distancia del sensor respecto a los vehículos, la distancia mínima desde la cuál se considera como nuevo vehículo, entre otros.

3.4. Configuración y puesta a punto

Como se ha mencionado anteriormente, para la recolección de datos se utilizó un sensor OAK-D con base en su capacidad para realizar tareas de Aprendizaje profundo trabajando con redes neuronales convolucionales. Pero también por su adaptabilidad a diferentes condiciones de iluminación. Este sensor, integrado con la Raspberry Pi 4 y colocados estratégicamente en la entrada del campus Chontalpa de la UJAT, capturó los datos en una resolución de 4K. La elección de esta alta resolución no solo se orientó hacia un posible uso de OCR para el reconocimiento de placas, sino también a la captura de detalles precisos que pueden ser significativos para la seguridad.

Ya con el sensor correctamente instalado e integrado con la Raspberry Pi 4 en la ubicación

correspondiente se procede a ejecutar el sistema. En la Figura 3.4 se puede observar el sensor OAK-D y la Raspberry Pi 4 ya integradas y colocadas en el soporte. Cuando un vehículo entra en la ROI, la información capturada por la OAK-D es procesada tanto por la misma OAK-D como por la Raspberry Pi 4. Dentro de la OAK-D se utiliza la biblioteca DepthAI y el detector MobileNetSSD para detectar y clasificar el vehículo. El algoritmo de seguimiento es ejecutado dentro de la Raspberry Pi 4, aquí se rastrea al vehículo a medida que se desplaza con base en las detecciones realizadas por MobileNetSSD. Con base en los resultados obtenidos por el algoritmo de seguimiento se realiza un conteo cuando los vehículos cruzan la línea de entrada. La Raspberry Pi 4 envía esta información a través de WiFi a la red de la universidad para su registro y posterior análisis.

Figura 3.4. Sensor OAK-D y Raspberry Pi 4 integrados e instalados en su soporte.

Capítulo 4

Experimentos y Resultados

En esta capítulo se describe el diseño experimental y los resultados obtenidos en el desarrollo de un sistema automático de detección y conteo de vehículos. Como se muestra en la Figura 4.1, se seleccionaron las ubicaciones para el sensor OAK-D y se ajustaron los parámetros críticos de MobileNetSSD y del algoritmo de seguimiento. Además, se evaluó el sistema utilizando las métricas adecuadas: *Precision* y *Recall* para la identificación automática y *Precision* para el conteo automático. Estas métricas fueron calculadas para cada una de las cuatro categorías de vehículos en diez muestras diferentes correspondientes a los diez experimentos realizados. Estos experimentos en condiciones naturales de iluminación permitieron demostrar su utilidad en condiciones reales en el campus Chontalpa de la UJAT.

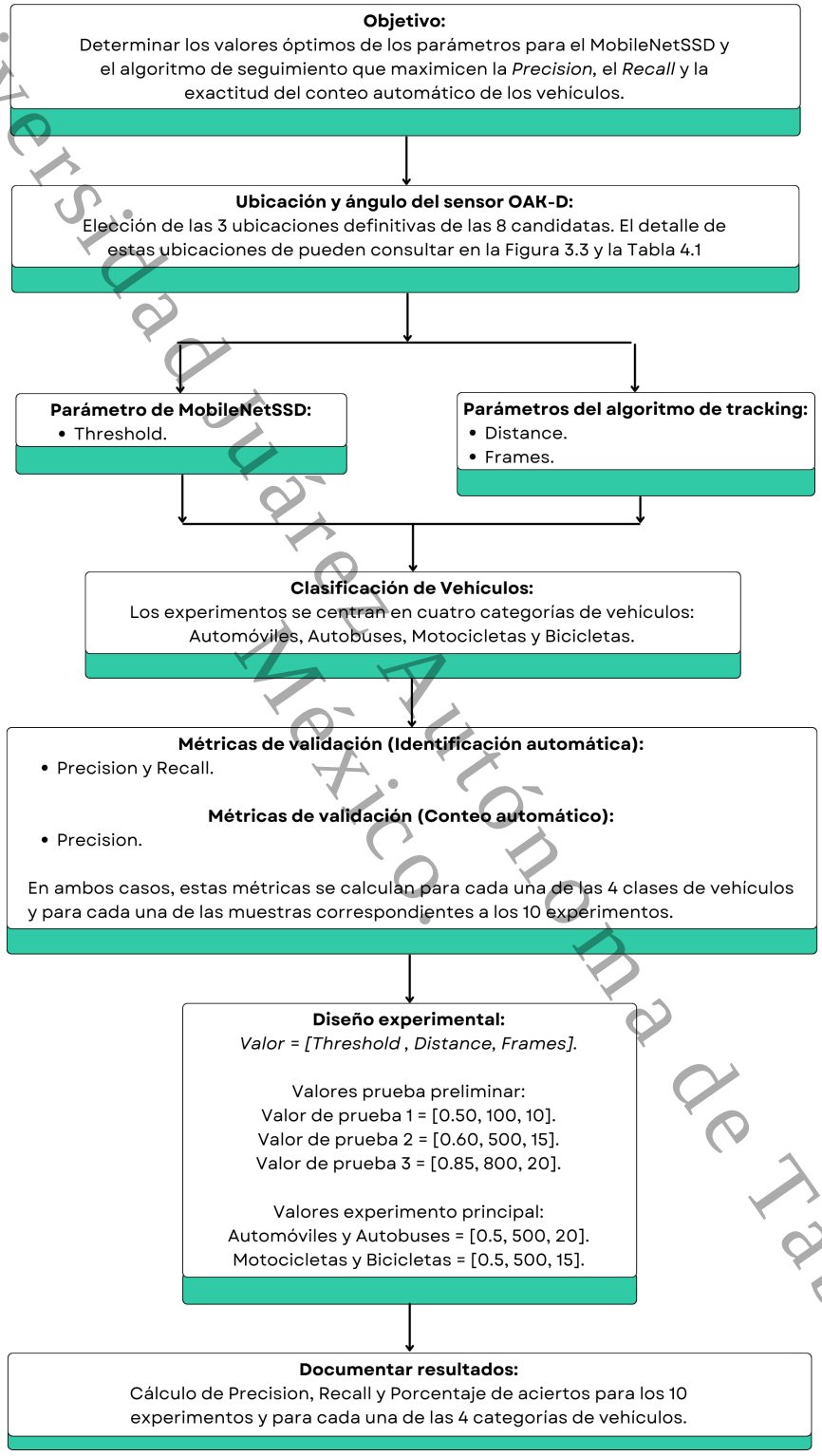


Figura 4.1. Diagrama del diseño experimental.

4.1. Diseño experimental

En esta sección se describe el diseño experimental. Se detalla el proceso de la ubicación del sensor, de la configuración de los parámetros y del proceso de recolección de datos. Además se explica el proceso de elección de los modelos de detección y de seguimiento de vehículos, así como de los métodos de evaluación de rendimiento del sistema.

Nuestro objetivo principal es apoyar en la logística y seguridad del campus utilizando tecnologías de Aprendizaje profundo, en específico un modelo de red neuronal convolucional, y una cámara OAK-D, respetando la privacidad y la confidencialidad de la información. Los experimentos se diseñaron con el fin de reflejar las condiciones reales en el campus durante todos los horarios de actividad, asegurando así la relevancia y aplicabilidad de nuestros resultados.

4.1.1. Ubicación del sensor

El sistema integrado se instaló en la entrada del campus Chontalpa de la UJAT en las posiciones mencionadas en la Sección 3.2. En la Figura 4.2 se muestra una toma de la entrada indicando las 3 ubicaciones del sensor que se utilizaron en el experimento principal (círculos numerados en la imagen) y se señala la ROI (*Region Of Interest* en color azul). Los componentes integrados son sujetados por un soporte para *smartphones* auto-adherente en la posición y en el ángulo correspondiente.

La posición se refiere a la altura a la que se coloca la cámara en relación con el suelo, mientras que el ángulo se refiere al grado en que la cámara apunta hacia el objetivo. La cámara a un nivel horizontal con respecto al horizonte corresponde a 0°, mientras que la cámara mirando hacia abajo se muestra con ángulos negativos. En la Tabla 4.1 se muestran los detalles de instalación del sensor OAK-D.

Tabla 4.1. Detalle de las tres ubicaciones del sensor para la captura de los datos.

Ubicación	Posición	Ángulo
①	1.8m	-15°
②	1.5m	-5°
③	1m	0°

Figura 4.2. Experimento principal en la entrada del campus Chontalpa de la UJAT usando la ROI y 3 posiciones diferentes del sensor OAK-D.

4.1.2. Configuración de parámetros

En esta etapa se llevó a cabo la preparación y el procesamiento de los datos capturados por el sensor OAK-D. Para adaptar las imágenes a los requisitos de la red neuronal MobileNetSSD, se redujo la resolución de las imágenes de alta calidad a 300x300 píxeles. Esta reducción de resolución permitió un procesamiento eficiente sin comprometer la calidad de la detección de los vehículos. En resumen, la preparación y el procesamiento de los datos aseguraron la adecuación de las imágenes capturadas para el análisis de detección de vehículos.

Ya en la ejecución práctica del sistema, se aplicó la red neuronal MobileNetSSD para detectar y clasificar vehículos. La clasificación se realizó en las categorías de automóviles, autobuses, bicicletas y motocicletas. La utilización de este modelo de red neuronal permitió una identificación precisa y eficiente de los vehículos en las imágenes procesadas. Después, se implementó

un algoritmo de *tracking* basado en distancia de centroides. Este algoritmo permitió realizar un seguimiento continuo de los vehículos a lo largo de los fotogramas y lograr el conteo.

Inicialmente, se realizaron pruebas preliminares modificando algunos parámetros de los elementos del sistema que están relacionados con distancias y umbrales de confianza. En la Tabla 4.2 se muestran los 3 diferentes valores para los parámetros del sistema durante las pruebas preliminares. El parámetro *threshold* de MobileNet indica el umbral de confianza de la red neuronal para predecir si un objeto dentro de una *bounding box* en particular pertenece a cierta clase. En cuanto al algoritmo de *tracking*, el parámetro *distance* indica la distancia entre los centroides de cada *frame* para identificar a un mismo objeto, mientras que el parámetro *frames* indica el número de veces que debe aparecer el objeto en la grabación para ser contado por el algoritmo de *tracking*.

Tabla 4.2. Configuración de los parámetros del sistema.

Elemento del sistema	**Parámetro**	**Valores de prueba**		
		Valor 1	**Valor 2**	**Valor 3**
`MobileNet`	*Threshold*	0.50	0.60	0.85
`Tracking`	*Distance*	100	500	800
`Tracking`	*Frames*	10	15	20

Derivado de las pruebas preliminares se obtuvieron los valores óptimos de los parámetros *threshold*, *distance* y *frames* mostrados en la Tabla 4.3. Tanto automóviles como autobuses comparten la misma configuración, lo mismo para motocicletas y bicicletas. El cambio en la configuración consiste en el algoritmo de *tracking*, el cual requiere cambiar sus parámetros *distance* y *frames* debido a que las bicicletas y las motocicletas son más pequeñas que un automóvil o autobús. Cabe mencionar que la ubicación del sensor no afecta en la configuración de parámetros seleccionada.

Tabla 4.3. Configuración de parámetros seleccionada.

		Automóvil, Autobús	**Motocicleta, Bicicleta**
Componente	**Parámetro**	**Valor**	**Valor**
`MobileNet`	*Threshold*	0.50	0.50
`Tracking`	*Distance*	500	200
`Tracking`	*Frames*	20	15

4.1.3. Captura de datos

Para la realización de experimentos se realizaron capturas durante el periodo del lunes 24 de abril al jueves 27 de abril de 2023. Además, se capturaron en diferentes horarios comenzando desde las 07:00 y finalizando a las 15:00 horas. Se buscó que hubiera diferentes condiciones climáticas, por ejemplo el lunes fue un día lluvioso, el martes fue un día particularmente nublado, mientras que el miércoles y el jueves fueron días despejados y soleados. Se buscó examinar las "horas pico" para tomar el mayor número de autos posible en el menor tiempo.

En la Tabla 4.4 se muestra el detalle de las 10 pruebas realizadas. La tabla incluye la ubicación del sensor en cada prueba, el día, la hora, la duración de la prueba, el estado del clima durante la prueba, y el número de vehículos que entraron durante ese periodo de tiempo (automóviles, motocicletas, autobuses y bicicletas).

Tabla 4.4. Detalle de las pruebas realizadas en los experimentos.

Prueba	Ubicación	Fecha	Hora	Duración	Clima	Total de vehículos
L1	2	24/abril	07:45	30 min	Lluvia	225
M1	3	25/abril	08:00	15 min	Nublado	123
M2	3	25/abril	14:00	10 min	Nublado	26
X1	2	26/abril	07:00	15 min	Soleado	19
X2	2	26/abril	09:00	10 min	Soleado	23
X3	2	26/abril	10:00	10 min	Soleado	42
X4	2	26/abril	11:50	10 min	Soleado	28
J1	3	27/abril	07:00	10 min	Soleado	21
J2	3	27/abril	07:35	25 min	Soleado	152
J3	1	27/abril	12:00	15 min	Soleado	37

4.2. Pruebas de detección

En primer lugar se presentan los resultados obtenidos por MobileNetSSD para la tarea de detección y clasificación de vehículos. Se expone una tabla para cada una de las categorías de vehículos (automóviles, motocicletas, autobuses y bicicletas). Por ejemplo, para la categoría automóviles véase la Tabla 4.5. Cada una de estas tablas indica el número real de vehículos que entran, además de las métricas *Precision* y *Recall* correspondientes a la clasificación de los

vehículos. Para el cálculo de estas métricas se utilizaron los siguientes conceptos de una matriz de confusión:

- TP (*True Positives*, verdaderos positivos): Ocurre cuando el detector reconoce un vehículo válido.
- FP (*False Positives*, falsos positivos): Ocurre cuando el detector identifica un objeto o el fondo de la escena como si fuera un vehículo cuando realmente no lo es.
- FN (*False Negatives*, falsos negativos): Ocurre cuando el detector no reconoce un vehículo válido.
- Nota: el valor de TN (*True Negatives*, verdaderos negativos) no aplica en el área de detección de objetos porque significa que el detector reconoce correctamente el fondo de un objeto como fondo, lo que equivale a no detectar nada.

En la Figura 4.3 se muestra un ejemplo del funcionamiento del detector MobileNet. De manera visual se despliega el *bounding box* sobre el objeto clasificando y se muestra el nombre de la clase y el porcentaje de confianza. Nótese que en el ejemplo ocurre un falso positivo para la clase Bicicleta y al mismo tiempo un falso negativo para la clase Motocicleta.

En la Tabla 4.5 se muestran los 10 experimentos realizados, haciendo énfasis únicamente en la clasificación de la clase Automóviles. En la primera columna se muestra el nombre del experimento. En la segunda columna se muestra el número de automóviles, mientras que en la tercera columna se muestra el número de aciertos obtenidos por el sistema. Es destacable el hecho de que en todos los experimentos se identificaron correctamente todos los vehículos, sin embargo, en la cuarta columna se pueden observar que hubo falsos positivos en 4 de los 10 experimentos (10 FP en total). En la penúltima columna se muestra el valor de la métrica *Precision*, la cual indica un valor de 1 para los casos en los que no hubo falsos positivos, y un valor menor a 1 en caso contrario. En la última columna se muestra el valor de la métrica *Recall*, el cual corresponde a 1 en todos los experimentos debido a que reconoce correctamente todos los automóviles reales.

Finalmente, en la parte inferior de la Tabla 4.5 se muestra el promedio de ambas métricas ($Precision_M$ y $Recall_M$).

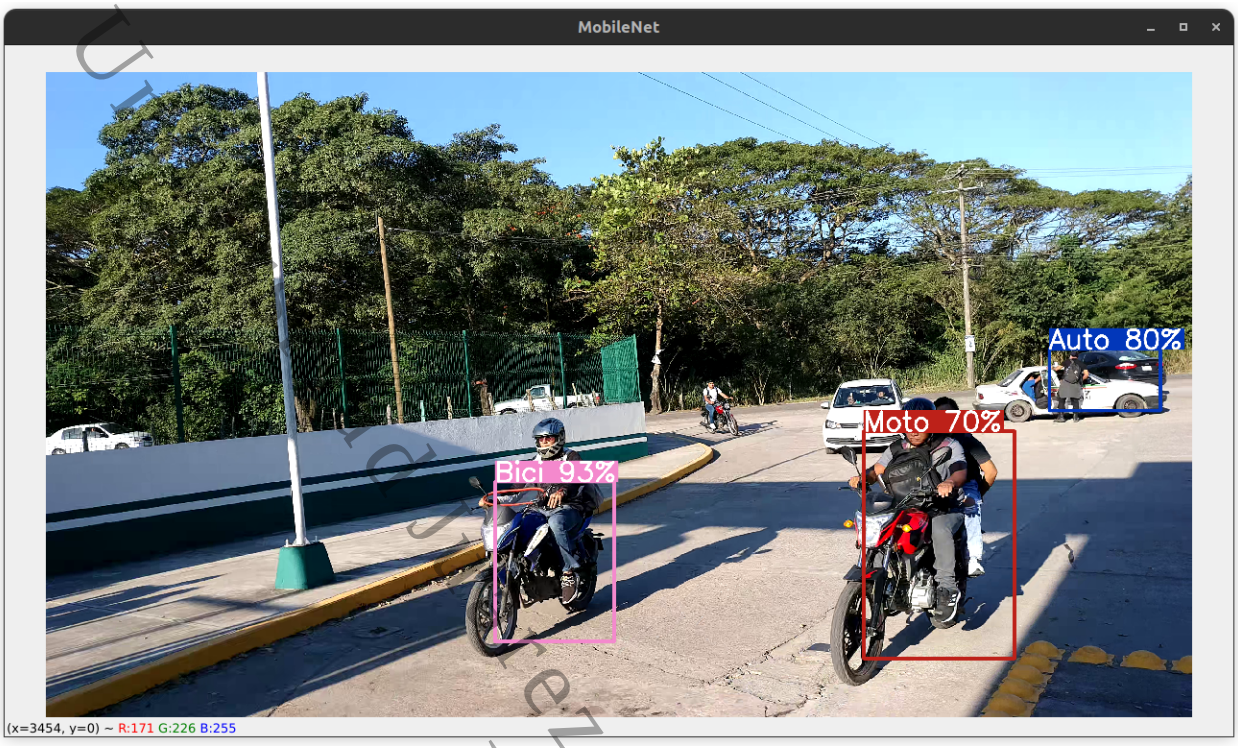


Figura 4.3. Ejemplos de detección de vehículos.

Para la categoría Motocicletas, la Tabla 4.6 detalla el desempeño del modelo MobileNetSSD en la clasificación de esta categoría. De igual manera que en la tabla anterior, la primer columna contiene los 10 experimentos. En la segunda y la tercer columna se exponen, respectivamente, la cantidad real de motocicletas que ingresaron al campus y la cantidad de detecciones correctas realizadas por el sistema. En estas cantidades podemos observar que no en todos los experimentos se detectaron y clasificaron correctamente el total de motocicletas.

Por ejemplo, en la prueba M1 se observaron 24 motocicletas en total. Sin embargo, el modelo solo obtuvo 22 TP. Este comportamiento puede ser reafirmado con los valores de la quinta columna donde se puede observar un total de 7 FN. Los FN representan, precisamente, las motocicletas que entran al campus pero no son detectadas por el sistema. La cuarta columna reporta el número de motocicletas detectadas por el sistema de manera errónea (FP). Es decir, los casos en que el sistema reporta una motocicleta pero no existe el vehículo o es de otra categoría. Para la columna correspondiente a la *Precision* se observan valores relativamente altos. Se tienen 7 valores iguales a 1 y 3 menores a 1 pero mayores a 0.9. En la última columna, correspondiente al *Recall* se muestran solo 5 valores iguales a 1 y los 5 restantes van desde 0.8 a 0.94. Finalmente,

Tabla 4.5. Métricas obtenidas por MobileNetSSD para la clase “Automóviles”.

Automóviles						
Prueba	**Valor real**	**TP**	**FP**	**FN**	**Precision**	**Recall**
L1	175	175	4	0	0.98	1
M1	97	97	1	0	0.99	1
M2	19	19	0	0	1	1
X1	14	14	0	0	1	1
X2	12	12	0	0	1	1
X3	24	24	0	0	1	1
X4	16	16	0	0	1	1
J1	13	13	0	0	1	1
J2	114	114	3	0	0.97	1
J3	25	25	2	0	0.93	1
			Promedio		**0.99**	**1**

en la fila final de la Tabla 4.6 se muestran los valores promedio para la *Precision* y el *Recall*.

Tabla 4.6. Métricas obtenidas por MobileNetSSD para la clase “Motocicletas”.

Motocicletas						
Prueba	**Valor real**	**TP**	**FP**	**FN**	**Precision**	**Recall**
L1	42	42	1	0	0.98	1
M1	24	22	0	2	1	0.92
M2	7	6	0	1	1	0.86
X1	5	4	0	1	1	0.80
X2	10	10	0	0	1	1
X3	13	13	0	0	1	1
X4	10	10	1	0	0.91	1
J1	7	6	0	1	1	0.86
J2	33	31	3	2	0.91	0.94
J3	12	12	0	0	1	1
			Promedio		**0.98**	**0.94**

En cuanto a los autobuses, la Tabla 4.7 presenta los resultados obtenidos por MobileNetSSD en su detección y clasificación. La tabla detalla los mismos 10 experimentos, mostrando de manera equivalente el conteo real de autobuses que ingresaron, los detectados correctamente (TP), los falsos positivos (FP) y los falsos negativos (FN). Es importante notar y mencionar que la cantidad de autobuses, en comparación con otros tipos de vehículos, es muy pequeña.

En 7 de los 10 experimentos no ingresa ni un solo autobús. En total, durante los 10 experimentos ingresaron 5 autobuses, de los cuales 4 fueron correctamente detectados y clasificados (TP). Esta frecuencia tan baja repercute directamente en la penalización que supone cada error de clasificación. Por ejemplo, en el experimento J2 no se detectó uno de los dos autobuses que entraron. Es solamente un falso negativo (FN), sin embargo, representa un fallo en el 50 % de los casos (2 autobuses en total). Debido a estos detalles, el modelo mostró una gran variabilidad en sus valores de *Precision* y *Recall* como puede observarse en las columnas 6 y 7. Finalmente, en estas mismas columnas, en la última fila se observan los bajos valores de *Precision* y *Recall* promedio obtenidos por el sistema.

Tabla 4.7. Métricas obtenidas por MobileNetSSD para la clase “Autobuses”.

Autobuses						
Prueba	**Valor real**	**TP**	**FP**	**FN**	**Precision**	**Recall**
L1	2	2	0	0	1	1
M1	1	1	1	0	0.50	1
M2	0	0	0	0	–	–
X1	0	0	0	0	–	–
X2	0	0	0	0	–	–
X3	0	0	0	0	–	–
X4	0	0	0	0	–	–
J1	0	0	0	0	–	–
J2	2	1	1	1	0.50	0.50
J3	0	0	0	0	–	–
			Promedio		**0.66**	**0.83**

Finalmente, la Tabla 4.8 analiza la capacidad del modelo en la identificación de bicicletas. De manera similar a las otras categorías, las primeras columnas de la Tabla muestran los 10 experimentos, el número real de bicicletas que ingresaron al campus (19 en total), las que fueron detectadas correctamente (17 de 19), los falsos positivos (FP, 10 en total) y los falsos negativos (FN, 2). De manera similar que con los autobuses, la cantidad de vehículos de clase bicicleta que ingresan al campus es relativamente pequeña. Esto se traduce como un efecto de volatilidad en las métricas *Precision* y *Recall*. Sin embargo, se pueden analizar algunos datos. Por ejemplo, con excepción de uno solo de los experimentos en los que sí ingresaron bicicletas, se puede observar la ocurrencia de FP. Esto se debe en gran medida a la confusión del modelo con las

motocicletas. Es decir, cuando una motocicleta entra a la región de interés y el modelo la clasifica incorrectamente como bicicleta. Esto repercute directamente en el rendimiento del sistema, dando como resultado una *Precision* notablemente más baja que su sensibilidad (*Recall*).

Tabla 4.8. Métricas obtenidas por MobileNetSSD para la clase "Bicicletas".

Bicicletas						
Prueba	**Valor real**	**TP**	**FP**	**FN**	**Precision**	**Recall**
L1	6	6	3	0	0.67	1
M1	1	1	0	0	1	1
M2	0	0	0	0	–	–
X1	0	0	0	0	–	–
X2	1	1	1	0	0.50	1
X3	5	5	3	0	0.63	1
X4	2	2	1	0	0.67	1
J1	1	1	1	0	0.50	1
J2	3	1	1	2	0.50	0.33
J3	0	0	0	0	–	–
			Promedio		**0.63**	**0.90**

Promediando el rendimiento de MobileNet en las 4 clases (automóviles, motocicletas, autobuses y bicicletas), tenemos una *Precision* global de 0.81 y un *Recall* global de 0.91.

4.3. Pruebas de *Tracking*

En esta sección se muestran los resultados del algoritmo del *tracking*, el cual depende de las detecciones obtenidas por MobileNet mostradas en la sección anterior. En este caso se muestra y evalúa el conteo realizado por el algoritmo durante el periodo de tiempo que duró cada experimento, clasificando cada una de las clases (automóviles, motocicletas, autobuses y bicicletas).

En la Figura 4.4 se muestran ejemplos del conteo de los automóviles, motocicletas, autobuses y bicicletas. De manera visual se despliega el *bounding box* sobre el objeto clasificando, el cual incluye un identificador temporal del objeto y el porcentaje de confianza. En la esquina superior izquierda se van contabilizando los vehículos por parte del *tracker*. En el ejemplo puede observarse que las camionetas son clasificadas como automóviles, y en este caso particular con un 99 % de confianza.

Figura 4.4. Ejemplo de conteo de vehículos.

En la Tabla 4.9 se muestran los 10 experimentos realizados enfocándose únicamente en la clase Automóvil. En la segunda columna se muestra el número total de automóviles que ingresaron al campus en ese periodo de tiempo. En la tercera columna se reporta VP, el número de

verdaderos positivos. Los verdaderos positivos representan el número de veces que el sistema identifica con éxito y agrega un nuevo vehículo el conteo. Es importante notar que en todos los casos los valores de VP coinciden con el número de vehículos real. Esto quiere decir que cada vehículo fue contado al menos una vez por el sistema. En la cuarta columna se muestra el número de vehículos contabilizados por el sistema. Esta cantidad es equivalente a la suma de los verdaderos positivos y los falsos positivos ($VP + FP$). Se puede observar que existe diferencia entre la cantidad de vehículos real contra la cantidad de vehículos contabilizada en 6 de los 10 experimentos. Sucede que el *tracker* pierde el conteo de *frames* de algún vehículo y eso provoca que lo cuente dos o más veces. En la última columna muestra el valor de la *Precision* del sistema en cada experimento. Finalmente, en la parte inferior de la Tabla 4.9 se muestra la *Precision* global del sistema.

Tabla 4.9. Resultados del conteo automático de la clase automóviles realizado con el algoritmo de *tracking*.

Automóviles				
Prueba	**Valor real**	**VP**	**VP+FP**	***Precision***
L1	175	175	183	0.96
M1	97	97	99	0.98
M2	19	19	20	0.95
X1	14	14	14	1.00
X2	12	12	12	1.00
X3	24	24	25	0.96
X4	16	16	18	0.88
J1	13	13	13	1.00
J2	114	114	115	0.99
J3	25	25	25	1.00
			Global	**0.97**

Con respecto a la clase motocicletas, la Tabla 4.10 despliega los resultados del algoritmo de *tracking* para esta categoría durante los 10 experimentos. Se muestran los valores correspondientes a la cantidad real de motocicletas que ingresaron al campus, los verdaderos positivos y el conteo realizado por el sistema. Observamos discrepancias entre la cantidad real de motocicletas y las reportadas por el sistema en todos los experimentos. Esto indica, que de manera similar que con los automóviles, el algoritmo contó algunas motocicletas más de una vez. Esto se debe a que el algoritmo a veces pierde el rastro de un vehículo y luego lo cuenta nuevamente. La última

columna indica la *Precision* obtenida por el sistema. En la última fila de la tabla se puede observar que la *Precision* global para las motocicletas es de 0.78. Este resultado sugiere que, aunque el sistema es generalmente eficaz en el seguimiento de motocicletas, es posible mejorar la precisión del conteo.

Tabla 4.10. Resultados del conteo automático de la clase motocicletas realizado con el algoritmo de *tracking*.

Motocicletas				
Prueba	**Valor real**	**VP**	**VP+FP**	***Precision***
L1	42	42	52	0.80
M1	24	24	27	0.88
M2	7	7	10	0.70
X1	5	5	7	0.71
X2	10	10	13	0.76
X3	13	13	21	0.68
X4	10	10	14	0.71
J1	7	7	9	0.77
J2	33	33	38	0.86
J3	12	12	15	0.80
			Global	**0.78**

En relación a los autobuses, los resultados presentados en la Tabla 4.11 exponen cómo el sistema manejó el conteo de estos vehículos durante los experimentos. De manera equivalente a los casos anteriores, en la tabla se presenta el número real de autobuses que entraron al campus, los verdaderos positivos, el conteo realizado por el sistema y el valor de *Precision* obtenido por el sistema. De igual forma, en la fila final de la tabla se encuentra el valor 0.83 que corresponde a la *Precision* global obtenida para la categoría autobuses. En este caso, solo en 3 de los 10 experimentos se registraron ingresos de autobuses al campus. Tomando en cuenta que el número limitado de autobuses se traduce a una volatilidad en los resultados, podemos realizar solo algunos comentarios generales. Por ejemplo, se puede observar que a los 5 autobuses que ingresaron al campus en total, el algoritmo de *tracking* le agregó un vehículo más. Es decir, el algoritmo estimó un total de 6 autobuses. Esto representa una proporción de 0.83 de precisión, cantidad que se muestra en la última columna en la fila final. Este 0.83 pareciera no ser malo, sin embargo, tampoco es suficiente para obtener un conteo de autobuses confiable. Como también se ha mencionado con las dos categorías de vehículos anteriores, el algoritmo de *tracking* a

veces tiende a realizar el conteo de cada autobús más de una vez.

Tabla 4.11. Resultados del conteo automático de la clase autobuses realizado con el algoritmo de *tracking*.

Autobuses				
Prueba	**Valor real**	**VP**	**VP+FP**	***Precision***
L1	2	2	2	1.00
M1	1	1	1	1.00
M2	0	0	0	–
X1	0	0	0	–
X2	0	0	0	–
X3	0	0	0	–
X4	0	0	0	–
J1	0	0	0	–
J2	2	2	3	0.67
J3	0	0	0	–
			Global	**0.83**

En cuanto a las bicicletas, la Tabla 4.12 revela cómo el algoritmo de *tracking* manejó el conteo de estos vehículos a lo largo de los 10 experimentos. La tabla compara el número real de bicicletas que ingresaron al campus, la cantidad que fue contada por el sistema y la cantidad de verdaderos positivos (VP). Se presentan diferencias en el conteo en 6 de los 7 experimentos en los que hubo ingresos de bicicletas. Los valores de *Precision* en los experimentos varían desde el 0.5 hasta un 1 en uno de ellos. Al igual que en las otras tres categorías de vehículos, el sistema tiende a contar dos o más veces una misma bicicleta. Esto resulta en una *Precision* global de 0.62 en el conteo automático de las bicicletas. Estos resultados indican que el sistema es capaz de seguir y contar bicicletas, pero por otro lado, existe la necesidad de mejorar su desempeño.

Promediando los resultados del algoritmo de seguimiento en las 4 clases (automóviles, motocicletas, autobuses y bicicletas), se obtiene una *Precision* global de 0.80.

Tabla 4.12. Resultados del conteo automático de la clase bicicletas realizado con el algoritmo de *tracking*.

Bicicletas				
Prueba	**Valor real**	**VP**	**VP+FP**	***Precision***
L1	6	6	10	0.60
M1	1	1	2	0.50
M2	0	0	0	–
X1	0	0	0	–
X2	1	1	2	0.50
X3	5	5	5	1.00
X4	2	2	3	0.50
J1	1	1	2	0.50
J2	3	3	4	0.75
J3	0	0	0	–
			Global	**0.62**

Capítulo 5

Contribuciones, conclusiones y trabajos futuros

5.1. Conclusiones

El presente trabajo se centró en un modelo de Aprendizaje profundo para la identificación y conteo automático de vehículos. En específico se utilizó un detector llamado MobileNetSSD basado en redes neuronales convolucionales. Respecto a la hipótesis planteada en la Sección 1.3, se ha demostrado que es posible alcanzar una precisión mayor al 80 % en la identificación y conteo automático de vehículos. En particular, el modelo de detección logró una *Precision* global de 0.81 y un *Recall* global de 0.91 en la detección de vehículos. Para el sistema integral de conteo autómatico se obtuvo una *Precison* global de 0.80, alcanzando el umbral establecido en la hipótesis. Por su parte, el objetivo general, mencionado en la Sección 1.4, de desarrollar un sistema automático de detección y conteo de vehículos, utilizando tecnología de Aprendizaje profundo y componentes de bajo costo se ha alcanzado. Esto se logró mediante la integración del modelo de detección MobileNetSSD, la implementación de un algoritmo de *tracking* que se basa en distancias, el uso de un sensor OAK-D y una placa Raspberry Pi.

Además, los objetivos específicos, mencionados en la Sección 1.5 también se cumplieron adecuadamente. Se logró la detección de vehículos según sus características físicas utilizando el modelo MobilenetSSD (véase sección 3.1). Se realizó el conteo automático de los vehículos

mediante un algoritmo basado en distancias de centroides (véase sección 3.1). Y finalmente, como se menciona en la sección 5.3, se produjo un reporte estadístico detallado con la información obtenida.

Sin embargo, aunque se alcanzó un rendimiento satisfactorio, existe la posibilidad de mejorar aún más el sistema. Se estima que con algunos ajustes adicionales, como agregar luz artificial o un reductor de velocidad, el rendimiento podría mejorarse hasta alcanzar o superar el 90 %.

En particular, el módulo detector logró identificar con un alto nivel de confianza a los automóviles y motocicletas, con una *Precision* de 0.99 y 0.98 respectivamente, así como una *Recall* de 1 y 0.94 respectivamente.

Sin embargo, el rendimiento del detector es bajo en cuanto a *Precision* para bicicletas y autobuses. Esto puede deberse a que hay pocos vehículos de estas 2 clases, es decir, sabemos que estas 2 categorías son las menos frecuentes en los experimentos. Debido a esto, las métricas de rendimiento tienden a ser dispersas de un experimento a otro. Por otro lado, sabemos que la *Precision* tiende a disminuir cuando el número de falsos positivos aumenta. Con base en esto y en adición al comportamiento observado en los experimentos, podemos identificar problemas puntuales con la *Precision*. Por ejemplo, para la clase bicicletas, muchos falsos positivos son reportados debido a confusiones con motocicletas (Figura 4.3). El *Recall* para autobuses fue relativamente alto, lo que indica que el modelo puede detectar la mayoría de los autobuses presentes. Sin embargo, y como ya se mencionó, presentó algunos errores reportando falsos positivos. En contraste, el *Recall* para bicicletas resultó más bajo. Esto demuestra que el modelo tiene problemas tanto para realizar una detección precisa como para evitar las omisiones para la categoría bicicletas. De nuevo, los problemas con este tipo de vehículos se deben a la confusión del detector con la clase motocicletas.

En cuanto al valor de *threshold* para el modelo MobileNetSSD, se observó un equilibrio entre una detección precisa y la minimización de errores. Un *threshold* demasiado bajo implica un aumento significativo de falsos positivos, donde objetos o el fondo son incorrectamente clasificados como vehículos. Por otro lado, un valor de *threshold* muy alto supone la posible omisión de vehículos reales, perdiendo *frames* importantes para el algoritmo de *tracking* y conteo automático. Por lo tanto, la selección de un *threshold* óptimo es crucial para equilibrar la precisión y la sensibilidad del sistema.

5.2. Trabajos futuros

Un área interesante para investigaciones futuras es la comparación y evaluación de otros modelos de detección, por ejemplo, YOLO y sus diferentes versiones (incluyendo TinyYOLO). Aunque en este trabajo se realizaron pruebas preliminares con estos modelos, fueron limitadas y poco formales. Se estima que, mediante la implementación de ajustes más avanzados y una configuración detallada, alguno de estos modelos podría proporcionar resultados competitivos. Por lo tanto, se propone realizar un análisis comparativo entre MobileNetSSD y YOLO al realizar la detección y clasificación de los vehículos dentro del sistema de conteo automático. Algunos de los puntos que podrían compararse son la precisión, la sensibilidad, la velocidad de procesamiento y la funcionalidad en diferentes condiciones. Este estudio comparativo permitiría determinar cuál de estos modelos es mejor para aplicaciones en el campo de la detección y conteo automático de vehículos.

Otro enfoque importante para futuros trabajos es la investigación y prueba de otros algoritmos de *tracking* más robustos. En este trabajo se eligió un algoritmo de *tracking* basado en distancias de centroides. Este algoritmo fue seleccionado principalmente por su simplicidad y bajo costo computacional en al realizar el seguimiento. A pesar de estos beneficios, este método tiene limitaciones, especialmente con vehículos a alta velocidad y por las condiciones variables de iluminación natural. Tomando en cuenta estas limitaciones, se propone evaluar algoritmos de *tracking* basados en Aprendizaje profundo. Esto con la intención de probar un enfoque mas complejo y sofisticado para el seguimiento de los vehículos. Estos modelos avanzados (por ejemplo DeepSORT) podrían mejorar la exactitud en el conteo y minimizar los errores. Sin embargo, es importante considerar el equilibrio entre la complejidad computacional y la eficacia del seguimiento. Por lo tanto, debe probarse si la implementación de *trackers* que utilizan técnicas de Aprendizaje profundo mejora significativamente la precisión del conteo sin entorpecer el rendimiento del sistema en términos de velocidad y eficiencia.

5.3. Creación del reporte estadístico

En esta etapa, se procedió a la creación de un reporte estadístico basado en los resultados obtenidos por el sistema de detección, clasificación, seguimiento y conteo automático de vehículos. El reporte proporciona información relevante y significativa que puede ser utilizada para la toma de decisiones y la planificación de actividades relacionadas con el control de acceso vehicular en el campus universitario. Algunos de los datos que se reportan son los siguientes:

- Número total de vehículos por día de la semana.
- Promedio de vehículos por día de la semana.
- Promedio de vehículos por categoría.
- Horas pico de tráfico.
- Distribución de vehículos por horarios de entrada.

Alojamiento de la Tesis en el Repositorio Institucional	
Título de la tesis:	Identificación automática de vehículos empleando aprendizaje profundo
Autor:	Fernando Contreras Pérez
ORCID:	`https://orcid.org/0000-0003-3882-2985`
Resumen:	Este trabajo de investigación se enfoca en la implementación y evaluación de un sistema de identificación, clasificación y conteo automático de vehículos utilizando tecnologías de Aprendizaje profundo. El principal objetivo de este sistema es el de mejorar el control de acceso vehicular en el campus Chontalpa de la Universidad Juárez Autónoma de Tabasco (UJAT). El sistema está compuesto por un sensor OAK-D y una Raspberry Pi para capturar, procesar y analizar los datos en tiempo real. El sistema propuesto, basado en la red neuronal MobileNetSSD, ha logrado una identificación de vehículos con una precisión global de 0.81 y una sensibilidad de 0.91. Este desempeño en la identificación influye positivamente en la tarea de conteo automático. El diseño experimental tomó en cuenta la recopilación y análisis de datos con diferentes configuraciones. Se ajustó el sistema a distintas condiciones de iluminación y climáticas. Las pruebas se realizaron en algunos puntos de la entrada del campus. También se variaron la ubicación y ángulos del OAK-D para mejorar la precisión de la detección. Este enfoque muestra que el sistema propuesto es adaptable a diferentes escenarios en condiciones naturales no controladas para entornos educativos. Todo esto, sin la necesidad de un equipo profesional costoso. El sistema propuesto ha demostrado ser una solución funcional y económica para el control de acceso vehicular, mejorando la seguridad y la eficiencia en el uso de recursos. Para investigaciones futuras, se propone explorar otros modelos de detección como YOLO así como algoritmos de seguimiento más complejos basados en Aprendizaje profundo. Estas mejoras podrían aumentar la precisión y la sensibilidad del sistema.
Palabras clave:	Inteligencia Artificial, Redes neuronales, Visión Computacional
Referencias citadas:	En la siguiente página se muestran las referencias.

Bibliografía

Ahmad, I. S. & Boufama, B. (2019). Automatic Vehicle Identification Through Visual Features. In *Proceedings of the 17th International Conference on Advances in Mobile Computing & Multimedia*, MoMM2019 (pp. 185–194). New York, NY, USA: Association for Computing Machinery.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1–6).: IEEE.

Aldea, E. (2017). *Raspberry PI fundamentos y aplicaciones*. Grupo Editorial RA-MA.

Aloysius, N. & Geetha, M. (2017). A review on deep convolutional neural networks. In *2017 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0588–0592).

Alzubi, J., Nayyar, A., & Kumar, A. (2018). Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series*, 1142, 012012.

Amit, Y., Felzenszwalb, P., & Girshick, R. (2020). *Object Detection*, (pp. 1–9). Springer International Publishing: Cham.

Azar, M., Tapia, Maria and Garc'ıa, J. L., & Pérez, A. (2019). Inteligencia artificial de las cosas. In *XXI Workshop de Investigadores en Ciencias de la Computación (WICC 2019, Universidad Nacional de San Juan).*

Berzal, F. (2019). *Redes neuronales & deep learning: Volumen II*. Independently published.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)* (pp. 3464–3468).

Blasch, E., Liu, S., Liu, Z., & Zheng, Y. (2018). Deep Learning Measures of Effectiveness. In *NAECON 2018 - IEEE National Aerospace and Electronics Conference* (pp. 254–261).

Chen, R.-C. (2019). Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning. *Image and Vision Computing*, 87, 47–56.

Chen, X., Xiang, S., Liu, C.-L., & Pan, C.-H. (2014). Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks. *IEEE Geoscience and Remote Sensing Letters*, 11(10), 1797–1801.

Cheng, G. & Han, J. (2016). A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117, 11–28.

Chibuluma, M. & Kalezhi, J. (2017). Application of a modified perceptron learning algorithm to monitoring and control. In *2017 IEEE PES PowerAfrica* (pp. 317–321).: IEEE.

Choudhary, R. & Gianey, H. K. (2017). Comprehensive Review On Supervised Machine Learning Algorithms. In *2017 International Conference on Machine Learning and Data Science (MLDS)* (pp. 37–43).

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. (2020). A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 27(4), 1071–1092.

Domingos, P. (2012). A Few Useful Things to Know about Machine Learning. *Commun. ACM*, 55(10), 78–87.

Elgendy, M. (2020). *Deep Learning for Vision Systems*. Manning Publications.

Elhoseny, M. (2020). Multi-object detection and tracking (MODT) machine learning model for real-time video surveillance systems. *Circuits, Systems, and Signal Processing*, 39, 611–630.

Everingham, M., Van Gool, L., Williams, C., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88, 303–338.

Freund, Y. & Schapire, R. (1996). Experiments with a new boosting algorithm. In *ICML*, volume 96 (pp. 148–156).: Citeseer.

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

George, A. & Marcel, S. (2020). Can Your Face Detector Do Anti-spoofing? Face Presentation Attack Detection with a Multi-Channel Face Detector.

George, A. & Marcel, S. (2021). Cross Modal Focal Loss for RGBD Face Anti-Spoofing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 7882–7891).

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Google AI (2018). Open Images Challenge. `https://storage.googleapis.com/openimages/web/challenge.html`.

Griese, M. & Kleinschmidt, J. (2019). Performance Analysis of a System for Vehicle Identification Using LoRa and RFID. In R. Miani, L. Camargos, B. Zarpelão, E. Rosas, & R. Pasquini (Eds.), *Green, Pervasive, and Cloud Computing* (pp. 115–127). Cham: Springer International Publishing.

Hao, X., Zhang, G., & Ma, S. (2016). Deep Learning. *International Journal of Semantic Computing*, 10(03), 417–439.

Hayman, S. (1999). The mcculloch-pitts model. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, volume 6 (pp. 4438–4439 vol.6).

Hnewa, M. & Radha, H. (2021). Object Detection Under Rainy Conditions for Autonomous Vehicles: A Review of State-of-the-Art and Emerging Techniques. *IEEE Signal Processing Magazine*, 38(1), 53–67.

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

IBM-Corporation (2021). Conceptos básicos de ayuda de CRISP-DM. `https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=dm-crisp-help-overview`.

Izidio, Diogo MF and Ferreira, A. & Barros, E. (2018). An Embedded Automatic License Plate Recognition System Using Deep Learning. In *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)* (pp. 38–45).: IEEE.

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31, 685–695.

Kaur, J. & Singh, W. (2022). Tools, techniques, datasets and application areas for object detection in an image: A review. *Multimedia Tools and Applications*, (pp. 1–55).

Krizhevsky, A., Sutskever, I., & Hinton, G. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM*, 60(6), 84–90.

Lao, R. (2017). Life of Data. Data Science is OSEMN. `https://medium.com/@randylaosat/life-of-data-data-science-is-osemn-f453e1febc10`.

Lau, C. H. (2019). Steps of a Data Science Project Lifecycle. *Towards Data Science*. *https://towardsdatascience.com/5-steps-of-adata-science-project-lifecycle-26c50372b492*.

Laurent, S. (2022). Machine Learning Algorithms: A Review. *Information Systems Journal*, ISJRA-3392, 6.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436–444.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

Li, Z., Dong, M., Wen, S., Hu, X., Zhou, P., & Zeng, Z. (2019). CLU-CNNs: Object detection for medical images. *Neurocomputing*, 350, 53–59.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. (2014). Microsoft COCO: Common Objects in COntext. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13* (pp. 740–755).: Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer Vision – ECCV 2016* (pp. 21–37). Cham: Springer International Publishing.

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR)*, 9, 381–386.

Mason, H. & Wiggins, C. (2010). A taxonomy of data science. `https://introdatasci.dlilab.com/pdf/A_Taxonomy_of_Data_Science.pdf`.

Minton, S. & Zweben, M. (1993). Learning, planning, and scheduling: An overview. *Machine Learning Methods for Planning*, (pp. 1–29).

Mondal, M., Mondal, P., Saha, N., & Chattopadhyay, P. (2017). Automatic number plate recognition using CNN based self synthesized feature learning. In *2017 IEEE Calcutta Conference (CALCON)* (pp. 378–381).: IEEE.

Montes, S. (2021). Desde tuberías arrancadas a miles de pesos en equipo electrónico: Las escuelas mexicanas son saqueadas durante la pandemia.

Padilla, R., Netto, S., & da Silva, E. A. (2020). A Survey on Performance Metrics for Object-Detection Algorithms. (pp. 237–242).

Parmar, P. & Tran Morris, B. (2017). Learning to Score Olympic Events. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Peralta, F. (2014). Proceso de conceptualización del entendimiento del negocio para proyectos de explotación de información. *Revista Latinoamericana de Ingeniería de Software*, 2(5), 273–306.

Perazzo, D., Soares, N., Lyra, V., Lima, G., Fontes da Gama, A., Xavier Natario Teixeira, J., & Teichrieb, V. (2022). OAK-D as a Platform for Human Movement Analysis: A Case Study. In *Symposium on Virtual and Augmented Reality*, SVR'21 (pp. 167–171). New York, NY, USA: Association for Computing Machinery.

Rahman, Z., Ami, A., & Ullah, M. (2020). A Real-Time Wrong-Way Vehicle Detection Based on YOLO and Centroid Tracking. In *2020 IEEE Region 10 Symposium (TENSYMP)* (pp. 916–920).

Ramírez-Sánchez, R. (2019). Del edén al infierno: inseguridad y construcción estatal en Tabasco. *LiminaR*, 17(2), 196–216.

Ravindran, R., Santora, M. J., & Jamali, M. M. (2021). Multi-Object Detection and Tracking, Based on DNN, for Autonomous Vehicles: A Review. *IEEE Sensors Journal*, 21(5), 5668–5677.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 28: Curran Associates, Inc.

Rojas-Perez, L. & Martinez-Carranza, J. (2021). Towards Autonomous Drone Racing without GPU Using an OAK-D Smart Camera. *Sensors*, 21(22), 7436.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., & Bernstein, M. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 211–252.

Saltz, Jeffrey y Suthrland, A. (2019). SKI: an Agile framework for data science. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 3468–3476).: IEEE.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Saoudi, O., Singh, I., & Mahyar, H. (2023). Autonomous Vehicles: Open-Source Technologies, Considerations, and Development.

Selmi, Z., Halima, M. B., & Alimi, A. M. (2017). Deep learning system for automatic license plate detection and recognition. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1 (pp. 1132–1138).: IEEE.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci*, 6(12), 310–316.

Shinde, P. P. & Shah, S. (2018). A Review of Machine Learning and Deep Learning Applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (pp. 1–6).

Singh Punn, N., Kumar Sonbhadra, S., & Agarwal, S. (2020). Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLOv3 and Deepsort techniques. *CoRR*, abs/2005.01385.

Spampinato, C., Palazzo, S., Giordano, D., Kavasidis, I., Lin, F.-P., & Lin, Y.-T. (2012). Covariance based Fish Tracking in Real-life Underwater Environment. In *VISAPP (2)* (pp. 409–414).

Tabasco Hoy (2022). Incrementa robo de autos en Tabasco: SESNSP. `https://www.tabascohoy.com/incrementa-robo-de-autos-en-tabasco-sesnsp/`.

Tang, K., Cao, Y., Chen, C., Yao, J., Tan, C., & Sun, J. (2021). Dynamic origin-destination flow estimation using automatic vehicle identification data: A 3D convolutional neural network approach. volume 36 (pp. 30–46).: Wiley Online Library.

Tang, Z., Naphade, M., Liu, M.-Y., Yang, X., Birchfield, S., Wang, S., Kumar, R., Anastasiu, D., & Hwang, J.-N. (2019). CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tourani, A., Shahbahrami, A., Soroori, S., Khazaee, S., & Suen, C. Y. (2020). A robust deep learning approach for automatic iranian vehicle license plate detection and recognition for surveillance systems. *IEEE Access*, 8, 201317–201330.

Vibha, L., Shenoy, P., Venugopal, K., & Patnaik, L. (2018). Moving vehicle identification using background registration technique for traffic surveillance. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1 (pp. 19–21).: IMECS.

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1 (pp. I–I).

Wageeh, Y., Mohamed, H., & Fadl, A. (2021). YOLO fish detection with Euclidean tracking in fish farms. *Journal of Ambient Intelligence and Humanized Computing*, 12, 5–12.

Wang, H., Yu, Y., Cai, Y., Chen, X., Chen, L., & Liu, Q. (2019). A comparative study of state-of-the-art deep learning algorithms for vehicle detection. *IEEE Intelligent Transportation Systems Magazine*, 11(2), 82–95.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)* (pp. 3645–3649).

Wu, B., Iandola, F., Jin, P. H., & Keutzer, K. (2017). SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Yang, W., Zhang, J., Wang, H., & Zhang, Z. (2018). A vehicle real-time detection algorithm based on YOLOv2 framework. In *Real-Time Image and Video Processing 2018*, volume 10670 (pp. 106700N).: International Society for Optics and Photonics.

Yang, X., Tang, Y. Y., Luo, H.-W., Wu, T., Sun, L., & Li, L. (2016). One sample based feature learning for vehicle identification. In *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 2 (pp. 1049–1054).

Yilmaz, A., Javed, O., & Shah, M. (2006). Object Tracking: A Survey. *ACM Comput. Surv.*, 38(4), 13–es.

Zhang, Y., Zhou, D., Chen, S., Gao, S., & Ma, Y. (2016). Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Álvarez Bazo, F., Sánchez-Cambronero, S., Vallejo, D., Glez-Morcillo, C., Rivas, A., & Gallego, I. (2020). A Low-Cost Automatic Vehicle Identification Sensor for Traffic Networks Analysis. *Sensors*, 20(19).