Diseño de Sistema de Monitoreo de Tránsito y Estacionamientos Vehiculares utilizando las Instalaciones de la Universidad Industrial de Santander como Laboratorio de Ciudad Inteligente.

Osmar Fabián Barón Torres y Julieth Vanessa Mejía Rodríguez

Tesis de grado para optar al título de Ingeniero Electrónico

Director

Efrén Darío Acevedo Cárdenas

Magíster en Ingeniería Electrónica

Codirector

Jaime Guillermo Barrero Pérez

Magíster en Potencia Eléctrica

Universidad Industrial de Santander

Facultad de Ingeniería Físico-Mecánica

Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2024

Agradecimientos

A mi madre Marlene, que ha sido mi motor y mi apoyo, sin ti esto no sería posible, gracias por creer en mí y por sacar a tus 3 hijos adelante sin importar los sacrificios.

En memoria de mi padre juan, quien sigue viviendo en mi corazón. En este momento tan importante tu ausencia se siente más que nunca, recuerdo como esperabas este logro con tanto orgullo, diciendo "Mi hija ya terminó materias y se va a graduar como Ingeniera". papito ahora puedo decir que lo logré. Aunque no estés físicamente aquí para presenciarlo, sé que estás sonriendo desde donde estés, orgulloso de cada paso que doy.

A mis hermanos Omer y Paola, no puedo imaginar mi vida sin ustedes, gracias por estar ahí cuando los he necesitado.

A mi cuñado Fabian, gracias por ser mi amigo y mi inspiración, gracias por tu ayuda incondicional, tu ejemplo ha sido una luz en mi camino.

A mi novio Juan, gracias por ser mi apoyo constante y por ser mi tutor, gracias por impulsarme a ser mejor cada día, eres mi mayor orgullo y mi mejor compañero de vida.

A mi mejor amiga Andrea, por acompañarme en cada trasnochada por ser mi hermana de vida y la mejor roomie.

A nuestro director y codirector de proyecto, su guía experta y su dedicación han sido fundamentales en este proceso. Gracias por su mentoría, por inspirarnos a alcanzar nuestras metas y por brindarnos las herramientas necesarias para hacer realidad este sueño académico.

Vane.

Que camino más largo y lleno de experiencias oscilantes entre momentos oscuros y sublimes. El verdadero logro no yace en haber llegado a este punto, el logro es haber seguido adelante sin apresurar el viaje, sin desfallecer y abrazando cada momento tanto bueno como malo. En la vida hay que valorar las pequeñas cosas de la vida ya que cuando nos demos cuenta y demos vuelta atrás, nos daremos cuenta de que esas eran las grandes.

Con gratitud y humildad, dedico este proyecto de grado a mi madre, mi padre, mis hermanos, mis tíos y mis amigos, cuyo apoyo ha sido esencial en mi travesía universitaria y en mi vida personal. Sus palabras de aliento y sus consejos han sido la fuerza motriz detrás de esta meta. A todos aquellos que han creído en mí y me han inspirado, les expreso mi más sincero agradecimiento.

Fabian.

Contenido

Pág.

Lista de Figuras

Pág.

Lista de Tablas

Pág.

Resumen

Título: Diseño de sistema de monitoreo de tránsito y estacionamientos vehiculares utilizando las instalaciones de la universidad industrial de Santander como laboratorio de ciudad inteligente[*]

Autores: Osmar Fabián Barón Torres y Julieth Vanessa Mejía Rodríguez[**]

Palabras Clave: Monitoreo de tránsito, estacionamiento vehicular, Ciudad inteligente, inteligencia artificial.

Descripción: En este proyecto se diseñó un sistema de monitoreo funcional y efectivo que permitió registrar y mantener un seguimiento preciso de la disponibilidad de espacios de estacionamiento en el campus de la Universidad Industrial de Santander, se empleó tecnologías como las redes neuronales convolucionales y la visión por computadora. Se optó como mejor opción el uso de la tarjeta Jetson Nano como herramienta de hardware, ya que cuenta con las características deseadas para este proyecto. En este caso, otorga potencia de cálculo para ejecutar modelos de IA y algoritmos de aprendizaje a bajo consumo de energía, como plataforma principal de software se usó Python, el cual se destaca por su simplicidad de sintaxis, lo que facilita la comprensión de código incluso para personas que son nuevas en la programación. Se logró desarrollar un sistema efectivo que automatiza la identificación de la capacidad vehicular en los espacios de estacionamiento dentro del campus universitario y se desarrolló una aplicación web para monitorear los resultados en tiempo real del sistema. Esta aplicación puede ser consultada por cualquier usuario si se tiene el enlace correspondiente.

Al considerar el campus universitario como un "laboratorio de ciudad inteligente", el proyecto también tiene como objetivo sentar las bases para futuros estudios que aborden el problema de congestión vehicular en la ciudad de Bucaramanga y su área metropolitana mediante herramientas tecnológicas.

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: Efrén Darío Acevedo Cárdenas, Magister en Ingeniería Electrónica. Codirector: Jaime Guillermo Barrero Pérez, Magíster en Potencia Eléctrica

Abstract

Title: Design of a traffic and parking monitoring system using the Universidad Industrial de Santander's campus as a smart city laboratory*

Authors: Osmar Fabián Barón Torres y Julieth Vanessa Mejía Rodríguez**

Key Words: Traffic monitoring, Vehicle parking, Smart city, Artificial intelligence.

Description: In this project, a functional and effective monitoring system was designed to register and maintain precise tracking of parking space availability on the campus of the Universidad Industrial de Santander. Technologies such as convolutional neural networks and computer vision were used. The Jetson Nano Developer Kit card was chosen as the best hardware tool as it has the desired characteristics for this project. In this case, it provides computing power to execute al models and learning algorithms whit low energy consumption. Python was used as the main software platform, which stands out for its simplicity of syntax, facilitating code comprehension even for those new to programming. An effective system was developed to automate the identification of vehicle capacity in parking spaces within the university campus, and a web application was developed to monitor real-time system results. This application can be accessed by any user with the corresponding link.

By considering the university campus as a "smart city laboratory", the project also aims to lay the groundwork for future studies that address the problem of vehicular congestion in the city of Bucaramanga and its metropolitan area using technological tools.

* Degree Work

** Faculty of Physical-Mechanical Engineering. School of Electrical, Electronic and Telecommunications Engineering. Director: Efrén Darío Acevedo Cárdenas, Master in Master in Electronic Engineering. Co-director: Jaime Guillermo Barrero Pérez, Master in Electrical Power

Glosario

Accuracy: precisión o accuracy en inglés, se refiere a qué tan aproximada es la predicción del modelo con respecto a los datos de entrenamiento suministrados.

Backend: es la parte de una aplicación que gestiona la lógica y el manejo de bases de datos, lo cual es invisible para los usuarios. Encargándose así del procesamiento, almacenamiento y manipulación de la información.

Batch normalization: esta capa normaliza lotes de datos antes de ingresarlos a las capas convolucionales. Su objetivo es estandarizar los datos para optimizar el proceso de entrenamiento mediante la convolución.

Batch size: es un hiper parámetro que determina la cantidad de imágenes que toma para realizar el entrenamiento simultáneamente. Este parámetro puede ser mayor dependiendo de los recursos computacionales con los que se cuente.

Capas: una capa es una agrupación de neuronas que representa una etapa del entrenamiento de la red. Existen varios tipos de capas las cuales tendrán un propósito específico.

Capa densa: son capas que procesan tensores de 1 sola dimensión y se usan con el fin de encontrar detalles específicos en la imagen.

Capa oculta: estas capas procesan información intermedia entre la capa de entrada y la de salida. Estas extraen características abstractas para identificar patrones y abordar tareas avanzadas.

Conexiones: las conexiones también llamadas sinapsis cuentan con pesos que indican la influencia en la salida. Los pesos más altos tienen mayor impacto, y los pesos más bajos, menor influencia.

Conv2D: una capa que crea un Kernel y lo convoluciona con la imagen de entrada para obtener características específicas de la imagen.

Dropout: es una técnica de regulación que consiste en establecer valores de los vectores en "cero", o en cierta medida desactivar neuronas intencionalmente para así evitar que el modelo empiece a aprenderse los valores.

Endpoints: un endpoint es un punto de acceso de los servicios web, utilizado para interactuar con el servicio a través de solicitudes HTTP, como obtener, enviar, actualizar o eliminar datos.

Épocas: una época hace alusión a un ciclo completo en el cual se ha ingresado los datos de entrenamiento a la red. Las épocas o "epochs" indican la cantidad de pasadas que realizan los datos de entrenamiento.

Flatten: esta capa se encarga de convertir un tensor de 3 dimensiones en uno de 1 sola dimensión. Esto con el fin de que los datos puedan ser usados en una capa densa posterior.

Frontend: es la parte visual de una aplicación con la que los usuarios interactúan directamente. Esta interfaz brinda la experiencia del usuario, utilizando HTML, CSS y JavaScript para entregar de manera organizada y comprensible la información.

HMI: hace referencia a la interfaz hombre-máquina, lo cual es el canal por medio del cual el software le entregará información al humano.

Kernel: es considerado el filtro que se aplica a una imagen para extraer ciertas características o patrones de esta. Este concepto en la estructura de las redes neuronales se refiere al tamaño que tendrá dicho filtro para la convolución.

MaxPooling2D: es una capa que busca reducir el tamaño de las imágenes a procesar con el fin de reducir la carga computacional e ignorar los pequeños cambios entre imágenes.

Neurona: la neurona es el elemento principal de las redes neuronales; la cual realiza operaciones matemáticas para transformar entradas en salidas. Estas crean conexiones con otras neuronas permitiendo la transmisión de información.

Overfitting: es un fenómeno que ocurre cuando un modelo se adapta en exceso a los datos de entrenamiento. La regularización controla pesos excesivamente grandes, mejorando la generalización a nuevos datos y evitando el overfitting.

Strides: es la distancia que determina cada cuántos píxeles se va a aplicar el filtro. La medida del stride determinará el tamaño de la salida de la convolución.

Tensores: son arreglos numéricos multidimensionales que pueden contar con 0, 1, 2, 3 o 4 dimensiones. Son la estructura básica de datos que se usa para deep learning.

Introducción

La zona urbana del Municipio de Bucaramanga y su Área Metropolitana están experimentando un rápido crecimiento que resulta en una congestión vehicular. Esta expansión, sumado a la falta de una red vial eficiente y sistemas de transporte público adecuados, así como al aumento constante de vehículos particulares, genera un tráfico más lento lo que impacta negativamente en la calidad del aire, la salud y los tiempos de viaje de los habitantes.

La estructura monocéntrica del Área Metropolitana de Bucaramanga tiene implicaciones importantes en la economía de la región. Por un lado, la concentración de actividades en un núcleo central puede generar ventajas en términos de economías de aglomeración, lo que puede impulsar el crecimiento económico y la competitividad. Sin embargo, también puede aumentar la congestión y los costos de transporte, ya que los trabajadores y consumidores deben desplazarse desde los municipios periféricos hacia el centro para acceder a los bienes y servicios.

A pesar de contar con un parque automotor de 849.000 vehículos (cifra del 2022, con un aumento de un 42% con respecto a 2015) (comunicaciones y prensa del área metropolitana de Bucaramanga, 12 de marzo de 2024), no se han logrado avances significativos en la construcción de infraestructuras capaces de hacer frente al aumento de la demanda. Las estadísticas indican que alrededor del 41% de las obras civiles en general presentan demoras. Además del tema de movilidad que claramente se ve afectado, estos proyectos generan sobrecostos que hacen que sea más difícil aún que se esté frecuentemente mejorando la infraestructura vial y civil.

Esto genera un desequilibrio que produce externalidades negativas del tráfico como congestión, ruido, polución, accidentes y estrés. Por ello, se hace necesario comenzar a abordar la problemática de la movilidad en Bucaramanga para proponer soluciones tecnológicas eficientes.

En el informe de la “Jornada de la movilidad sostenible y promoción de modos de transporte sostenible” de septiembre del 2018 se analizó la efectividad de las herramientas tecnológicas para el análisis de las vías del municipio de Bucaramanga. Para esta pequeña validación se tomaron datos de velocidad promedio para el sector comprendido desde la Cra. 9 hasta la Cra.19 y desde la Calle 32 hasta la Calle 43.

Este estudio arrojó resultados muy interesantes porque validan la precisión y efectividad de las herramientas tecnológicas, como Google maps para este caso. Y a pesar de que se utilizó exitosamente datos de velocidad promedio para el análisis de la implementación de las medidas restrictivas de “Pico y placa”, no se ha vuelto a acudir a la tecnología para tratar de darle un respiro a la ciudad de Bucaramanga en temas de movilidad.

En este sentido, es fundamental que se tome ventaja de estas tecnologías para lograr mejoras significativas en la movilidad y en la calidad de vida de la sociedad en general. En lugar de limitarnos a soluciones convencionales y tradicionales, es necesario que se incorporen herramientas tecnológicas innovadoras que nos permitan solucionar los problemas de movilidad de manera más efectiva.

Es por ello por lo que se ha decidido dar un paso más en dirección a la modernización de las redes viales, controlando y supervisando la “capacidad” que tiene un entorno como la Universidad Industrial de Santander. Se considera un punto de inicio clave, ya que la variable de tránsito “capacidad” suele ser la que estipula los límites de movilidad que tiene una vía y de cierta manera establece los puntos donde comienza a generar conflictos viales.

Con fines prácticos, hemos optado por evaluar esta "capacidad" como la cantidad de vehículos que el campus puede alojar en términos de espacios de estacionamiento. Esto se logra mediante una monitorización en tiempo real de la entrada y salida de vehículos, lo que nos permite

conocer la disponibilidad de espacios o la capacidad en el campus en cada momento del día. Esta iniciativa tiene un objetivo adicional: el desarrollo de una herramienta de fácil acceso para el personal de seguridad de la universidad. De esta manera, se les proporciona acceso a estos datos, lo que les permite supervisar estas áreas sin tener que desplazarse a los distintos estacionamientos del campus o llevar a cabo un registro manual.

Asimismo, como un beneficio adicional, el personal administrativo tendrá acceso a una base de datos que les permitirá consultar los registros de flujos vehiculares diarios en la entrada de la institución. Esto agilizará la obtención de información crucial en caso de que sea necesaria para tomar decisiones relacionadas con proyectos de construcción de vías y planeación.

Este proyecto pretende cimentar los fundamentos para una investigación futura y la implementación de sistemas tecnológicos basados en inteligencia artificial, orientados hacia el campo de la movilidad y el transporte. De esta manera, se establece un primer paso en el estudio de estas aplicaciones y su contribución al proceso de mejora de la movilidad en el área metropolitana de Bucaramanga.

1. Objetivos

1.1 Objetivo General

Diseñar un sistema inteligente ubicado estratégicamente dentro del campus universitario que permita identificar el tipo de vehículos y la cantidad de vehículos que ingresan y salen de las instalaciones de la universidad industrial de Santander. Además, determinará los cupos de parqueo disponibles y las variables de tránsito necesarias, dicha información se subirá a una base de datos que podrá ser consultada en línea.

1.2 Objetivos Específicos

Realizar un análisis de la capacidad y ubicación de los estacionamientos de la Universidad Industrial de Santander para actualizar una base de datos en la cual se podrá visualizar la cantidad de espacios disponibles para estacionar los vehículos.

Utilizar procesamiento de imágenes y redes neuronales para identificar los diferentes tipos de vehículos, si ingresan o salen de las instalaciones, el flujo vehicular y su respectivo conteo.

Utilizar y complementar una base de datos que permita entrenar una red neuronal convolucional para identificar tipos de vehículos y su sentido de circulación.

Crear una base de datos que pueda ser consultada en línea por medio de un aplicativo que contenga información como el número de vehículos presentes en el campus y la cantidad de parqueaderos disponibles.

Programar un sistema embebido para realizar pruebas de campo y enviar los resultados a una base de datos.

2. Marco Teórico

Este proyecto se desarrolló en un entorno tecnológico que abarca desde la visión artificial hasta las redes neuronales convolucionales. En este contexto, el marco teórico actual tiene como objetivo proporcionar una comprensión sólida de los fundamentos esenciales para entender su aplicación específica en este proyecto.

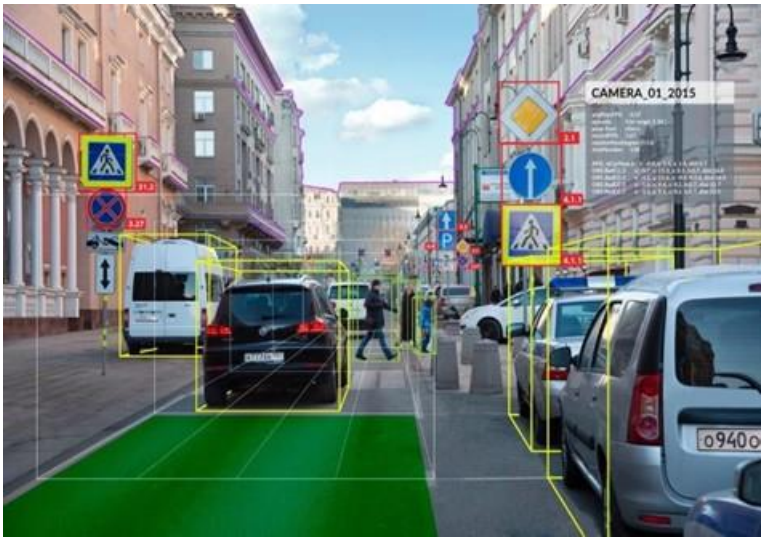
Se analizaron conceptos clave, como el aprendizaje profundo, las redes neuronales convolucionales y las técnicas de procesamiento de imágenes, con el propósito de establecer las bases teóricas que respaldan el desarrollo y la exitosa implementación de soluciones avanzadas en el ámbito de la movilidad y el control vehicular.

Visión artificial: La tecnología de visión artificial se centra en capacitar a las máquinas para interpretar su entorno a través de imágenes y videos. Su aplicación abarca desde el reconocimiento de objetos en tiempo real en vehículos autónomos hasta el análisis de imágenes médicas para el diagnóstico. La identificación de objetos es un uso común, beneficiando sectores como seguridad, automatización industrial y mejoras en infraestructuras de transporte.

Además, la visión por computadora se emplea en la detección de rostros y seguimiento de objetos, así como en sistemas de vigilancia en tiempo real mediante cámaras de seguridad para detectar personas o vehículos, analizando movimientos y patrones. Un ejemplo de esto se puede observar en la figura 1.

Figura 1.

Identificación de vehículos



Nota: Gavilán, I.G.R. (16, octubre, 2020). *Principales aportaciones de la visión artificial*. [Anónimo]. https://ignaciogavilan.com/principales-aportaciones-de-la-vision-artificial/

Inteligencia Artificial (IA): La inteligencia artificial es un campo de la informática que busca crear sistemas capaces de realizar tareas que normalmente requerirían intervención humana, como razonamiento, aprendizaje y resolución de problemas. Se basa en algoritmos y modelos matemáticos para permitir a sistemas automatizados aprender de datos, adaptarse y tomar decisiones inteligentes. La IA abarca áreas como el aprendizaje automático, procesamiento del lenguaje natural, visión por computadora y robótica. El aprendizaje automático destaca al desarrollar algoritmos para que las máquinas aprendan sin programación explícita, mientras que el procesamiento del lenguaje natural se centra en que las máquinas comprendan el lenguaje humano, y la visión por computadora les permite interpretar imágenes y videos.

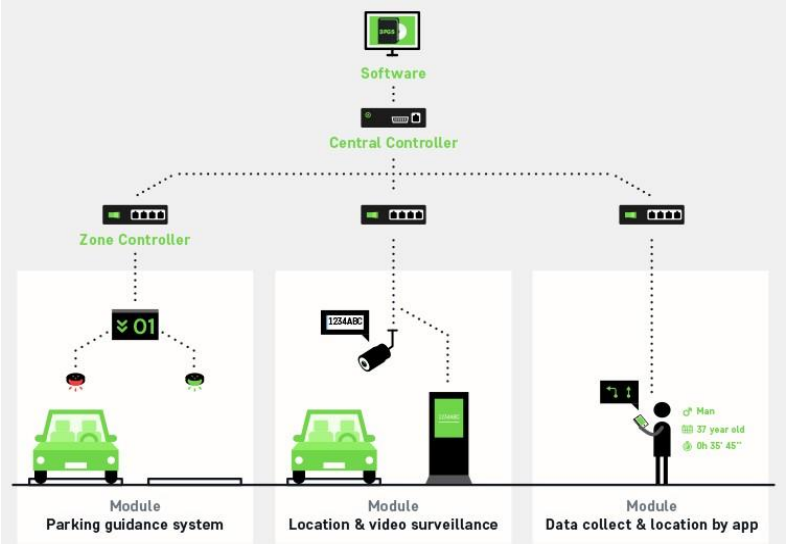
Sistema de Orientación de Estacionamiento (SOE) o Parking Guidance System (PGS): es una solución que proporciona información en tiempo real sobre la disponibilidad de espacios de estacionamiento en áreas de aparcamiento. Estos sistemas, comúnmente equipados con

indicadores luminosos LED, dirigen a los usuarios hacia lugares libres, evitando la búsqueda caótica y congestiones innecesarias. Por ejemplo, se puede observar una representación visual de este concepto en la figura 2. Además de ofrecer datos en tiempo real, los SOE pueden proporcionar información sobre la cantidad total de vehículos y espacios disponibles, facilitando la administración del parqueadero y guiando a los conductores hacia espacios convenientes. Estos sistemas versátiles se aplican en diversos entornos de alta afluencia vehicular como centros comerciales, aeropuertos y universidades.

Los datos recopilados por el SOE no solo informan sobre la disponibilidad de estacionamiento, sino que también son valiosos para analizar patrones de uso y demanda a lo largo del tiempo. Estos datos respaldan la planificación y proyección de modificaciones futuras en las instalaciones del estacionamiento, contribuyendo a decisiones informadas para mejorar eficiencia y rentabilidad.

Figura 2.

Sistema guía de parqueo



Nota: McKellar, J. (2021). Parking Guidance Systems. https://blog.nortechcontrol.com/parking-guidance-system)

Aprendizaje automático: El aprendizaje automático se utiliza comúnmente para prever patrones o resultados en grandes conjuntos de datos, siendo beneficioso en áreas como la medicina y el comercio electrónico. Se basa en la capacidad de un software o dispositivo para aprender de forma autónoma, empleando algoritmos y modelos matemáticos para extraer información clave de los datos. El proceso consta de tres etapas: aprendizaje, entrenamiento y generación de resultados. El modelo examina datos, identifica patrones en la fase de aprendizaje, se ajusta durante el entrenamiento y luego se utiliza para producir resultados, permitiendo tomar decisiones fundamentadas en diversas aplicaciones.

Aprendizaje Supervisado: El aprendizaje supervisado implica entrenar un modelo con datos previamente etiquetados, proporcionando al modelo un conjunto de datos de entrada junto con las salidas esperadas (etiquetas). El modelo aprende a establecer relaciones entre los datos de entrada y las salidas esperadas con el objetivo de crear una función que pueda predecir resultados para nuevas entradas no vistas previamente. Este enfoque utiliza un conjunto de datos de entrenamiento con etiquetas para capacitar al modelo, que luego se emplea para realizar predicciones sobre nuevos datos sin etiquetas.

Aprendizaje No Supervisado: En el aprendizaje no supervisado, un modelo de inteligencia artificial se entrena para descubrir patrones y relaciones en los datos sin la guía explícita de etiquetas. A diferencia del aprendizaje supervisado, no se utilizan respuestas predefinidas, centrándose en la extracción de información significativa de los datos. Algunas técnicas incluyen la agrupación, la reducción de dimensionalidad, la detección de anomalías y la asociación de reglas.

Aprendizaje por Refuerzo: El aprendizaje por refuerzo se centra en cómo un agente de inteligencia artificial puede aprender a tomar decisiones óptimas a través de la interacción con su

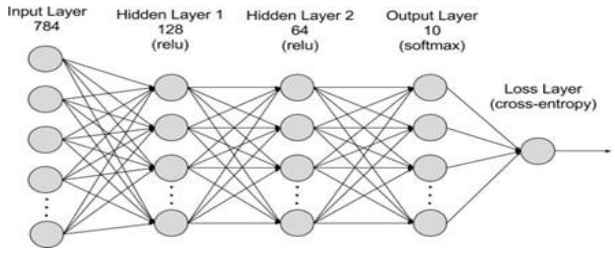
entorno. El agente aprende mediante la experiencia, recibiendo recompensas o castigos según sus acciones en un entorno específico, con el objetivo de maximizar la recompensa acumulada a lo largo del tiempo. Aprende de sus errores y ajusta su comportamiento en consecuencia.

Redes Neuronales: Las Redes Neuronales son modelos computacionales inspirados en la estructura cerebral humana, destacando por su capacidad para procesar información mediante la interconexión de millones de neuronas. Su característica clave es el aprendizaje a partir de datos de entrada, ajustando los pesos sinápticos mediante algoritmos de aprendizaje para optimizar el rendimiento. Con aplicaciones en visión por computadora, procesamiento del lenguaje natural, robótica, diagnóstico médico y predicción financiera, estas redes son versátiles y adaptables para abordar problemas complejos. Una ilustración de este proceso se puede observar en la figura 3 que muestra cómo las neuronas se interconectan entre sí a lo largo de las capas, formando una red.

A diferencia de la programación convencional, no dependen de reglas predefinidas, sino que reciben entradas y generan salidas mediante procesamiento a través de capas interconectadas de neuronas. La intrigante capacidad radica en que, a diferencia de los algoritmos con programación explícita, las redes neuronales pueden aprender y ajustar sus pesos sinápticos para generar salidas precisas sin comprender las reglas o la lógica subyacente de manera explícita.

Figura 3.

Redes Neuronales



Nota: Amazon Web Services, Inc. (2023). *¿Qué es una red neuronal? - Explicación de las redes neuronales artificiales – AWS.* [Anónimo]. https://aws.amazon.com/es/what-is/neural-network/

Entrenamiento de la red: El entrenamiento de una neurona y sus conexiones para alcanzar pesos y sesgos óptimos implica exponer la red a ejemplos que guíen su aprendizaje. Se utiliza un conjunto de ejemplos llamado conjunto de datos o *dataset,* con valores iniciales aleatorios para pesos y sesgos. Tras calcular las salidas, la red evalúa su rendimiento y ajusta gradualmente los pesos y sesgos según los resultados. Cada ciclo de entrenamiento aporta información numérica, permitiendo a la red extraer características clave del problema. Este proceso iterativo es crucial para afinar los parámetros de la red y mejorar su capacidad predictiva.

Aumento de datos: Una estrategia sumamente beneficiosa radica en expandir la base de datos de entrenamiento mediante la inclusión de nuevos datos generados a partir de transformaciones aplicadas a los datos originales. Este enfoque permite abarcar un espectro más amplio de ejemplos de entrenamiento. En el contexto del procesamiento de imágenes, por ejemplo, es posible llevar a cabo transformaciones como rotaciones, cambios de tamaño, variaciones de posición y ajustes de color sobre los datos iniciales, lo que resulta en la creación de una base de datos de entrenamiento más sólida y abarcadora.

Función de activación: La función de activación es un marco matemático que optimiza la aproximación de una función requerida, permitiendo predecir sus salidas según las entradas. En su ausencia, las neuronas operan linealmente, limitando su capacidad de resolver diversos problemas. Las funciones de activación superan esta limitación al permitir a las redes neuronales abordar problemas más complejos, como la identificación de sonidos, textos e imágenes, al introducir no linealidades en el proceso.

Unidad lineal rectificada (RELU): La función ReLU (Rectified Linear Unit) se posiciona como una de las funciones más ampliamente utilizadas en el ámbito de las redes neuronales, destacando por su mayor velocidad de cálculo y su menor consumo de recursos computacionales.

A diferencia de otras funciones, ReLU no establece un límite superior en los valores de salida, ya que puede generar valores desde '0' hasta el infinito. Sin embargo, esta característica conlleva un desafío conocido como "neuronas muertas". Este fenómeno se manifiesta cuando varias neuronas en una red configurada con la función ReLU quedan atrapadas en valores de '0', lo que obstaculiza significativamente el proceso de aprendizaje.

$$max\ (0, x)$$

Softmax: La función Softmax se emplea principalmente en problemas de clasificación, con el propósito de proporcionar una respuesta más precisa en relación con la decisión final de una red neuronal en tareas de clasificación. Esta función permite obtener resultados más esclarecedores en términos de las decisiones tomadas por la red neuronal de clasificación.

$$f(Z)_j = \frac{e^{Zi}}{\sum_{k=1}^{k} e^{zk}}$$

Dropout: es una herramienta que permite alternar el uso de diferentes neuronas en una red neuronal para aliviar la carga de pesos que algunas conexiones acumulan en exceso. Durante el entrenamiento, se desactivan ciertas neuronas según una configuración predefinida, generalmente a través de una probabilidad expresada en porcentaje. Esto equilibra el uso de todas las neuronas y conexiones, resultando en una red más resiliente. Análisis de imágenes en redes neuronales: Cuando se aborda el análisis de imágenes con el propósito de desarrollar modelos de entrenamiento, es esencial adoptar un enfoque distinto al utilizado en las redes neuronales convencionales. En estas últimas, suele emplearse características específicas como entrada, vinculadas al problema a resolver. Sin embargo, al tratar imágenes, se requiere abordar la tarea de manera diferente. Cada imagen debe descomponerse en píxeles, siendo cada uno de estos píxeles una característica de entrada para la red neuronal. Esto confiere una gran robustez a los sistemas de clasificación de imágenes, pero al mismo tiempo exige una mayor capacidad de recursos

computacionales. Para ilustrar esto, una sola imagen en una resolución estándar de 720x480 puede constar de 345,600 píxeles, aunque esto no es perceptible para el ojo humano, salvo que la imagen se encuentre en una resolución más baja en la que se pueden distinguir cada uno de los pixeles, como se puede ver en la figura 4.

Es común que, con el objetivo de reducir la carga computacional y optimizar el proceso de entrenamiento, se analicen imágenes en blanco y negro. La conversión de una imagen a escala de grises, donde los valores oscilan entre '0' y '255', puede resultar una estrategia efectiva, siempre que el problema a resolver no dependa del color. No obstante, subsiste el desafío de enseñar a la red neuronal que un píxel no constituye una característica aislada, ya que, si se trata de identificar un objeto con diferentes orientaciones, tamaños o ubicaciones, la red podría no reconocerlo debido a las variaciones en los valores de los píxeles. Para abordar esta dificultad, entran en juego las redes convolucionales.

Figura 4.

Imagen dividida en pixeles

Nota: Trackglobe. (2024). *Google mejora la experiencia con imágenes pixeladas.* https://www.trackglobe.com/general/google-mejora-la-experiencia-con-imagenes-pixeladas

Redes Neuronales Convolucionales: Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) representan una categoría especializada de redes neuronales diseñadas

específicamente para tareas avanzadas de clasificación y detección de imágenes. La arquitectura de estas redes introduce dos tipos de capas adicionales: la capa de convolución y la capa de agrupación.

El concepto clave en la convolución de imágenes radica en analizar cada píxel considerando las variaciones o diferencias con los píxeles adyacentes. En este proceso, se extrae un núcleo que consiste en una matriz de píxeles seleccionados para la convolución, junto con valores ponderados asignados a cada uno de estos píxeles en el núcleo. Luego, se aplican operaciones específicas para calcular el nuevo valor del píxel principal.

Este proceso produce una nueva imagen con características distintas, conocidas comúnmente como filtros en la edición de imágenes (como desenfoques, solarización, afilado, realce de contornos, etc.).

En las capas de convolución, se ejecutan operaciones con múltiples núcleos, generando una cantidad equivalente de imágenes transformadas, cada una con diferentes tipos de características. Estas transformaciones son elegidas de manera automática por la red, de manera similar a cómo se ajustan los pesos de las conexiones en una red neuronal convencional.

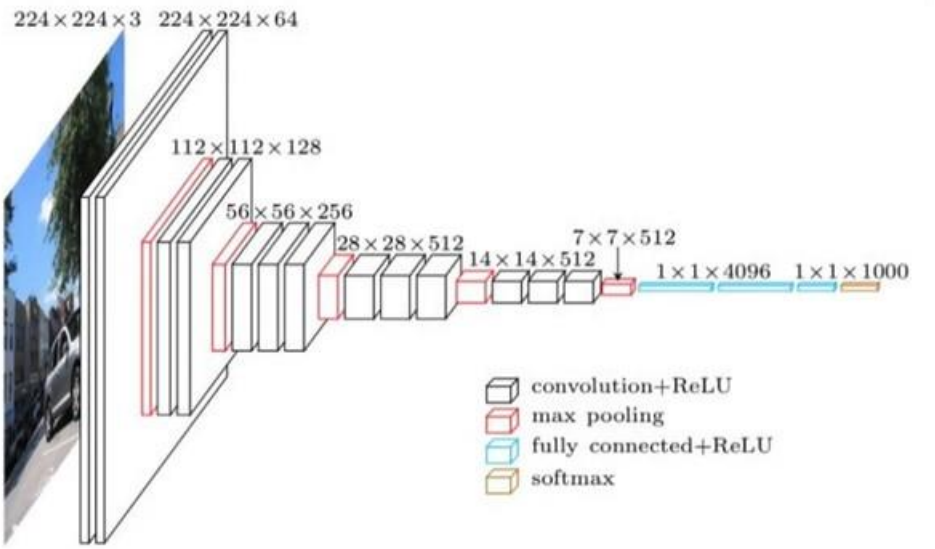
En el caso de imágenes en color, cada núcleo se descompone en tres subnúcleos, correspondientes a los canales de color rojo, verde y azul. Se realizan convoluciones separadas para cada canal y luego se suman los resultados para obtener el núcleo final.

Las capas de agrupación utilizan núcleos con un desplazamiento ajustable conocido como "kernel" para determinar cómo se mueven los núcleos entre los píxeles. Esta operación de agrupación selecciona el píxel con el valor más alto para resaltar las características más relevantes de la imagen. Como resultado, se obtiene una imagen de menor tamaño en comparación con la original, pero con las características más destacadas resaltadas.

A medida que las imágenes son procesadas por cada una de las capas, sus dimensiones van cambiando, dependiendo del tipo de capa que la procese. Este cambio se puede observar en la figura 5.

Figura 5.

Dimensiones de las capas en un modelo



Nota: Sotaquirá, M. (23, marzo, 2019). Codificando Bits. ¿Qué son las Redes Convolucionales? https://www.codificandobits.com/blog/redes-convolucionales-introduccion/

Arquitectura Alex Net: Esta arquitectura de red neuronal fue diseñada específicamente para reconocimiento de imágenes y aporta un concepto bastante útil, el cual es la variación del tamaño del Kernel a medida que se avanza en las capas convolucionales. La primera capa cuenta con un Kernel de (11,11) con la intención de captar características más generales de la imagen.

En las siguientes capas se empieza a disminuir el Kernel a (5,5) y (3,3) para poder captar a detalle todas las particularidades de la imagen mediante los filtros convolucionales. Cada una de estas capas va seguida de una capa de Maxpooling2D.

Por otro lado, se adiciona una capa de Batch Normalization, la cual se encarga de acelerar

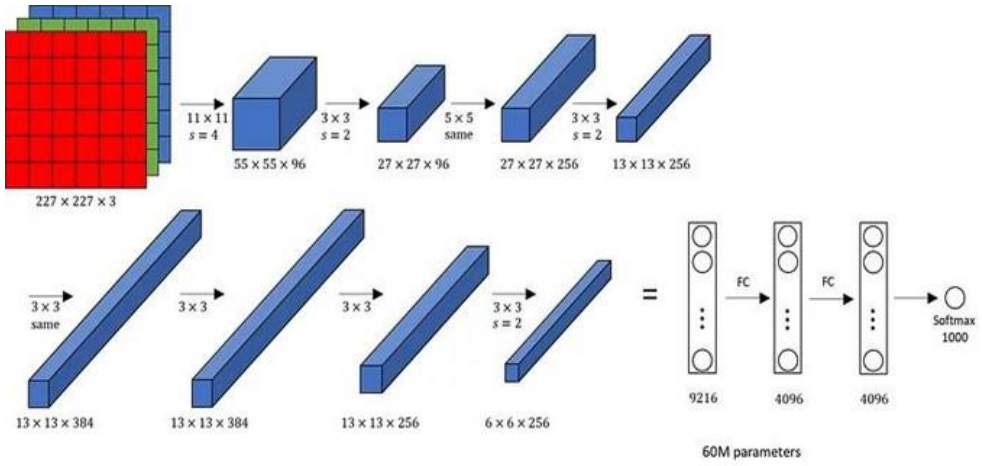
el entrenamiento y de mejorar la capacidad del modelo para generalizar características.

Este tipo de capas es usado en redes profundas, las cuales necesitan extraer características de objetos que se encuentran en ambientes poco controlados.

Por último, se añade una capa de *Flatten*, la cual es usada con el fin de 'aplanar' los datos que vienen de una capa que entrega un vector multidimensional, convirtiéndolos en un vector de una sola dimensión para que pueda ser procesado adecuadamente en una capa densa de neuronas, las cuales complementan el modelo para llegar a una salida más precisa. El la figura 6 se puede observar cómo va evolucionando el tamaño de los datos a medida que pasa por las capas de la arquitectura AlexNet.

Figura 6.

Estructura de capas en un modelo AlexNet.



Nota: Bangar, S. (Jun 24, 2022). The convolutional neural network (CNN) architecture known as AlexNet was created by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, who served as Krizhevsky's PhD advisor. https://medium.com/@siddheshb008/alexnet-architecture- explained-b6240c528bd5

Herramientas de Software

Tensor Flow: TensorFlow es una biblioteca de código abierto para aprendizaje automático que se centra en redes neuronales artificiales y modelos de aprendizaje profundo. Utilizable en C++ y Python, ofrece capacidades como la creación de gráficos de flujo de datos, construcción de estructuras de redes, definición de nodos y ejecución en diversos entornos.

Gestiona las conexiones entre entradas y salidas, liberando al desarrollador de detalles complejos y simplificando el proceso de desarrollo.

Tensorboard: Para analizar los resultados del proceso de entrenamiento de las redes neuronales, se cuenta con una herramienta sumamente útil conocida como "Tensorboard". Esta herramienta proporciona una interfaz gráfica que permite observar en tiempo real el rendimiento de una red a medida que avanza a través de las épocas de entrenamiento.

En términos generales Tensorboard nos entrega dos gráficas principales: la precisión versus épocas y la pérdida versus épocas. Estas gráficas nos brindan información esencial que facilita la evaluación del proceso de entrenamiento de la red.

Keras: es una biblioteca de código abierto en Python que simplifica el desarrollo de redes neuronales y se integra con TensorFlow. Proporciona una interfaz de programación consistente para crear prototipos eficientes tanto en CPU como en GPU. Keras es altamente personalizable, permitiendo la adaptación fácil a las necesidades específicas del proyecto.

Es compatible con redes neuronales convolucionales y recurrentes, brindando versatilidad para abordar diversas aplicaciones de aprendizaje automático. Además, facilita la colaboración y el intercambio de modelos y capas entre profesionales.

OPEN CV: es una biblioteca de código abierto esencial para la visión por computadora y aprendizaje automático. Destacándose en el procesamiento en tiempo real de imágenes y videos,

se utiliza en la detección de objetos, segmentación de imágenes, seguimiento y calibración de cámaras.

Compatible con Windows, Linux, Android e iOS, y adaptable a C++, Python y Java, ofrece una amplia gama de funciones para procesamiento de imágenes, detección de características, seguimiento de objetos y reconstrucción en 3D. Su versatilidad y robustez la convierten en una opción primordial para aplicaciones en visión por computadora.

Hardware

Kit de desarrollo NVIDIA Jetson Nano: La Jetson Nano de NVIDIA es una placa de desarrollo compacta diseñada para aplicaciones en aprendizaje automático e inteligencia artificial. A pesar de su tamaño, puede ejecutar redes neuronales complejas, siendo adecuada para tareas como clasificación de imágenes, detección de objetos y procesamiento de voz. Presentada en marzo de 2019, cuenta con una GPU NVIDIA Maxwell de 128 núcleos, un procesador de cuatro núcleos quad-core ARM Cortex-A57 y 4GB de RAM LPDDR4. Es una opción económica y eficiente en consumo energético (5 a 10 vatios) para desarrolladores, investigadores y entusiastas de la IA, con un rango de precios que oscila entre los US$ 200 - US$ 250. Ofrece conectividad fácil con puertos estándar, como GPIO, USB 3.0, HDMI y Ethernet.

Compatible con marcos de trabajo populares como TensorFlow, PyTorch y Keras, la Jetson Nano incluye abundante documentación en línea para facilitar la iniciación en el desarrollo.

Tabla 1.

Jetson Nano

	Especificaciones técnicas
Jetson Nano.	**GPU:** Maxwell de 128 núcleos **CPU:** ARM A57 de 4 núcleos @143Ghz **Memoria:** 4 GB 64-bit LPDDR4 de 25.6GB/s  **Almacenamiento:** No cuenta con almacenamiento interno, pero sí con espacio para microSD Card. **Codificador de video:** 4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265) **Decodificador de video:** 4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265) **Conectividad:** Ethernet de hasta 1GB/s, M.2 Key E **Display:** HDMI y puerto para display **USB:** 4x USB 3.0, USB 2.0 Micro-B **Otros:** GPIO, I2C, I2S, SPI, UART **Alimentación:** 5 vdc

3. Análisis Instalaciones de la UIS

3.1 ¿Por qué se usa la UIS como laboratorio de ciudad inteligente?

Según el plan de desarrollo institucional 2019 -2030 de la Universidad Industrial de Santander, declara:

> Se estimulará la inversión privada en modernización y aprovechamiento de tecnologías productivas y de inteligencia (Internet de las Cosas - IoT, analítica de datos, inteligencia artificial, sistemas autónomos), se plantean las bases para una política satelital, se co-financiará la transformación digital territorial pública e iniciativas de ciudades inteligentes (Universidad Industrial de Santander, 2019, pág. 27).

La introducción de tecnología puede ser clave para el progreso de la ciudad y la solución de diversos problemas. Como miembros de la Universidad Industrial de Santander, tenemos el deber de aportar al desarrollo regional mediante investigadores con impacto social positivo.

Para alcanzar este objetivo, es vital realizar pruebas graduales en entornos controlados, comenzando por la monitorización de la capacidad vehicular en el campus universitario. Esto permitirá gestionar eficazmente la circulación vehicular en los parqueaderos de las instalaciones, proporcionando metodologías para diseñar soluciones replicables en la ciudad.

Dado el bajo flujo vehicular y las dimensiones limitadas del campus, es ideal para analizar patrones de tráfico. Es fundamental abordar el concepto de “Capacidad”, que determina cuántos vehículos pueden circular sin afectar la movilidad. Se evalúan los espacios de estacionamiento como parte de este análisis, y se ha desarrollado una herramienta de gestión que podría beneficiar a los encargados de administrador estos espacios.

Figura 7.

Mapa de la Universidad Industrial de Santander.

Nota: Universidad Industrial de Santander [Anónimo]. (2024). Mapa Universidad Industrial de Santander. https://uis.edu.co/mapa/

Figura 8.

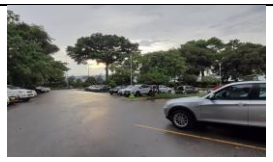
Plano Universidad Industrial de Santander.

Parqueaderos dentro del Campus

En la tabla 2 se detalla la capacidad vehicular de cada uno de los parqueaderos de la universidad, acompañada de una fotografía real y su ubicación en los planos de la Universidad Industrial de Santander.

Tabla 2.

Parqueaderos del campus.

Parqueadero	Capacidad	Ubicación (Plano)	Foto Real
Parqueadero adyacente al edificio de Alta Tensión (P7)	Autos: 44 Motos: 50	 Parqueadero P7 en los planos de la UIS.	 Parqueadero P7 tomada por los autores.
Parqueadero Fisicomecánicas (P8)	Autos: 10	 Parqueadero P8 en los planos de la UIS.	 Parqueadero P8 tomada por los autores.
Parqueadero del edificio Jorge Bautista Vesga (P9)	Autos: 7	 Parqueadero P9 en los planos de la UIS.	 Parqueadero P9 tomada por los autores.

Parqueadero	Capacidad	Ubicación (Plano)	Foto Real
Parqueadero Planta de Aceros (P10)	Autos: 4	 Parqueadero P10 en los planos de la UIS.	 Parqueadero P10 tomada por los autores.
Parqueaderos en la vía detrás de “El Bosque”	Autos: 44	 Parqueadero de “El Bosque” en los planos de la UIS.	 Parqueadero de “El Bosque” tomada por los autores.
Capacidad total:		Autos:109 Motos: 50	

Nota: La tabla cuenta con imágenes de los planos y fotos reales de los parqueaderos del campus a considerar. Además, cuenta con la capacidad de cada uno de ellos.

Consideraciones:

En el marco de este proyecto se consideraron exclusivamente las zonas de parqueo que están destinadas para los vehículos que ingresan a través de la entrada de la Carrera 25 con 8va. Ya que estos vehículos tienen restringidas las zonas en las cuales se les permite estacionar.

En la práctica se excluyeron las siguientes zonas de estacionamiento: parqueaderos del estadio, parqueadero de visitantes, parqueadero exclusivo de administrativos y el parqueadero del auditorio Luis A. Calvo.

Es preciso aclarar que para el desarrollo del proyecto los espacios de parqueo de las bicicletas no se tendrán en cuenta ya que al ser un vehículo tan ligero y versátil puede estacionarse

casi en cualquier lugar del campus. Sin embargo, sí se realizó la identificación y conteo de este tipo de vehículo.

Figura 9.

Entrada de la 25 #8va en los planos de la UIS.

4. Diseño de la Red Neuronal

Para el desarrollo es esta etapa se llevó a cabo la siguiente metodología:

- Creación del Data Set:

Como primer paso se construyó un dataset con imágenes de vehículos (carros, motos y ciclas) obtenidas de diferentes páginas web con licencia “Creative Commons”, lo cual se refiere a que tienen un Copyright que permite hacer uso de las imágenes sin necesidad de pedir permiso al autor.

Aumento de datos: Para poder lograr un modelo con una precisión óptima, es esencial disponer de una base de datos sólida que permita un entrenamiento efectivo de la red. Una técnica

sumamente valiosa es el aumento de datos, el cual permite ampliar la base de datos de entrenamiento mediante la inclusión de nuevos datos generados a partir de transformaciones aplicadas a los datos originales. De esta manera, se abarca un espectro más amplio de información de entrenamiento. Para expandir nuestra base de datos, se emplearon transformaciones como rotaciones, ajustes de tamaño, variación de posiciones y alteración del zoom o acercamiento, generando imágenes a partir de cada imagen original.

Figura 10.

Imagen original sin data augmentation

Figura 11.

Imagen nueva con data augmentation.

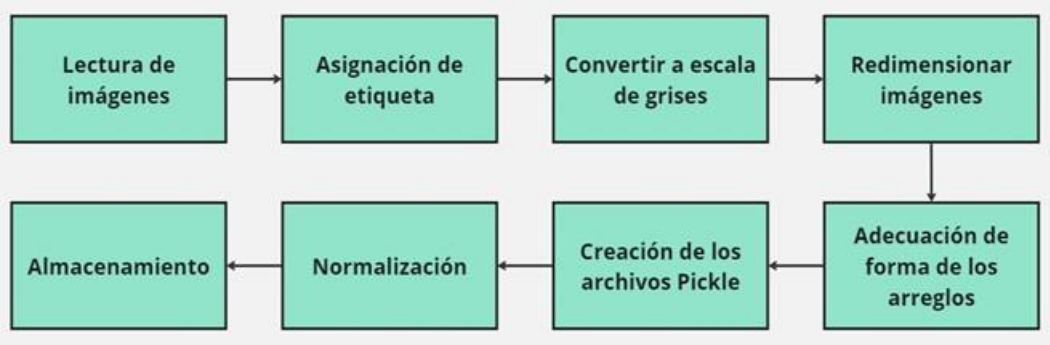
Nota: Se puede observar que la imagen se desplazó unos pixeles a la derecha y hacia arriba, de la misma manera tuvo una alteración en su dirección horizontal.

Con esta herramienta, nuestra base de datos pasó de 1551 imágenes a un total de 10129 imágenes entre carros, motos y bicicletas. Sin embargo, es fundamental usar esta técnica con prudencia, ya que un uso excesivo puede provocar un sobreajuste (Overfitting) en el modelo. El paso a paso del tratamiento de las imágenes para el aumento de datos se describe en la figura 12.

A partir de este punto se procedió con la generación de las secuencias de datos para que puedan ser leídos por la red neuronal para su entrenamiento.

Figura 12.

Pasos para realizar el aumento de datos o data augmentation.



Al descargar las imágenes, se llevó a cabo su clasificación en las categorías de vehículos relevantes, que incluyen automóviles, motocicletas y bicicletas. Cada una de estas categorías se organizó en directorios específicos con el propósito de etiquetarlas para el entrenamiento de la red neuronal.

Posteriormente, se realizó un paso para simplificar el procesamiento de las imágenes. Se eliminaron los componentes de color RGB de las imágenes, dejando únicamente un canal en escala de grises en lugar de tres canales.

Esto se hizo para reducir la carga de procesamiento, ya que, en este contexto el color de los vehículos no es relevante, y el enfoque está en estudiar sus formas.

A continuación, se ajustó la dimensión de cada elemento para estandarizar la cantidad de

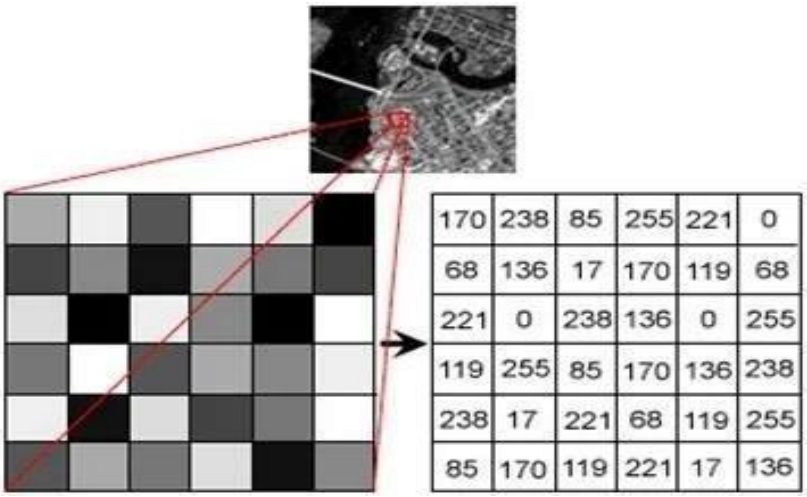
píxeles en cada imagen. La base de datos original contenía elementos de diferentes tamaños y formatos, por lo que era fundamental que la entrada para la red neuronal tuviera una forma específica con dimensiones predefinidas (ancho=200px, alto=100px).

Para simplificar aún más el proceso para la red neuronal, se normalizaron los valores de los píxeles. En una escala de grises, los valores oscilan originalmente entre 0 (negro) y 255 (blanco), cubriendo así toda la escala de grises como se puede ver en la figura 13, donde se evidencian los valores que puede adquirir una imagen en escala de grises. Después de la normalización, estos valores oscilan en un rango de 0 a 1.

Luego, se crearon archivos Pickle para almacenar de manera serializada la información de cada imagen, lo que facilita su manejo y procesamiento posterior por parte de las librerías de entrenamiento de redes como Tensorflow.

Figura 13.

Representación matricial de una imagen en escala de grises.



Nota: Datasmarts (sf). Cómo manipular los pixeles de una imagen con Opencv. https://www.datasmarts.net/como-manipular-los-pixeles-de-una-imagen-con-opencv/

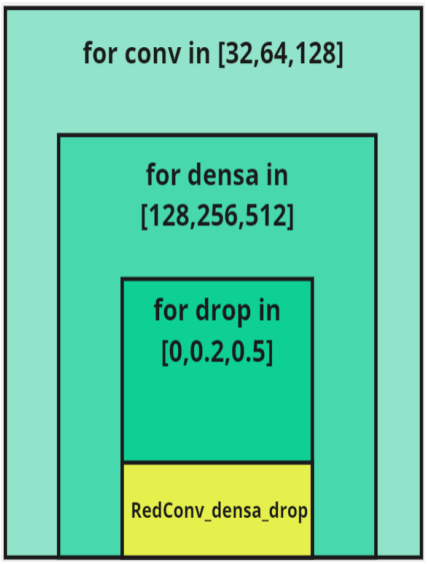
Selección del modelo: Para la selección del modelo se llevaron a cabo pruebas entre diferentes arquitecturas con el fin de compararlas y así escoger la que entregue mejores prestaciones con el menor costo computacional, ya que a pesar de que la tarjeta Jetson Nano ofrece buen desempeño en temas de procesamiento de imágenes, también es inteligente reducir la carga debido al alto uso que tendría la red a lo largo del tiempo.

Arquitectura básica de red neuronal convolucional: Como primera opción se contempló una red sencilla que cuenta con los elementos más básicos de una red neuronal convolucional como se describe a continuación:

Esta red cuenta con 2 capas convolucionales “Conv2D” con sus respectivas capas de MaxPooling2D cada una. Adicional una capa Densa con activación Relu y una capa de Dropout. Con la intención de analizar diferentes configuraciones se varió los parámetros de cada una de las capas anteriores de la siguiente manera:

Figura 14.

Representación gráfica del anidado de los “For”.



Estas diferentes combinaciones se usaron en el código de entrenamiento con un arreglo de "for" anidados para así obtener un comparativo de los diferentes modelos que se generan al combinar dichas configuraciones; con el fin de evaluar diferentes cantidades de filtros, neuronas en la capa densa y valores de dropout, logrando así escoger la mejor combinación de valores para esta arquitectura. Como resultado se obtuvieron 27 redes neuronales diferentes basadas en la mezcla de los parámetros de Dropout, Neuronas de las capas densas y los filtros de las capas convolucionales.

Tabla 3.

Ejemplos de 8 de las 27 redes neuronales generadas por los "for" anidados.

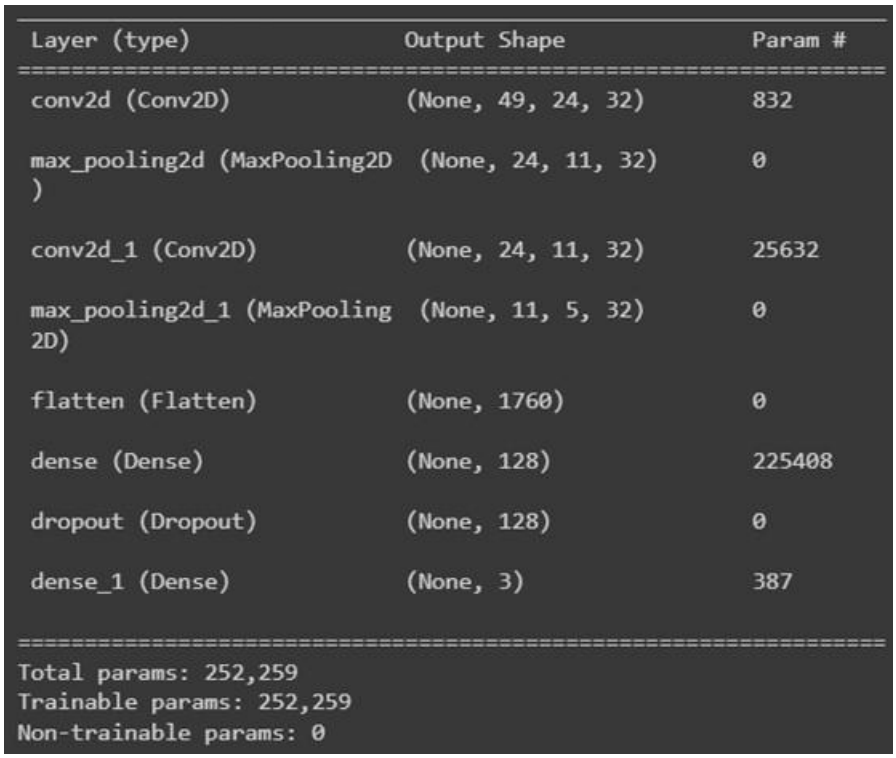
Red	Capa Conv. 1	Capa Conv. 2	Capa Densa	Dropout
Red_ejemplo1	32	32	128	0
Red_ejemplo2	32	32	128	0.2
Red_ejemplo3	32	32	128	0.5
Red_ejemplo4	32	32	256	0
Red_ejemplo5	32	32	256	0.2
Red_ejemplo6	32	32	256	0.5
Red_ejemplo7	32	32	512	0
…	…	…	…	…
Red_ejemplo27	128	128	512	0.5

Nota: En la anterior tabla se puede ver la estructura de 8 de las 27 redes con la misma arquitectura, pero con diferencia en los valores de sus parámetros. Siendo los valores de "Capa Conv." la cantidad de filtros convolucionales por capa, en los valores de "Capa Densa" la cantidad de neuronas por capa y además el correspondiente al "Dropout".

A continuación, se muestra un ejemplo del resumen del entrenamiento de una de estas redes, donde se puede apreciar cada capa y los parámetros entrenables que se desprenden de cada una de las capas.

Figura 15.

Resumen de un ejemplo de arquitectura de un modelo de red neuronal.



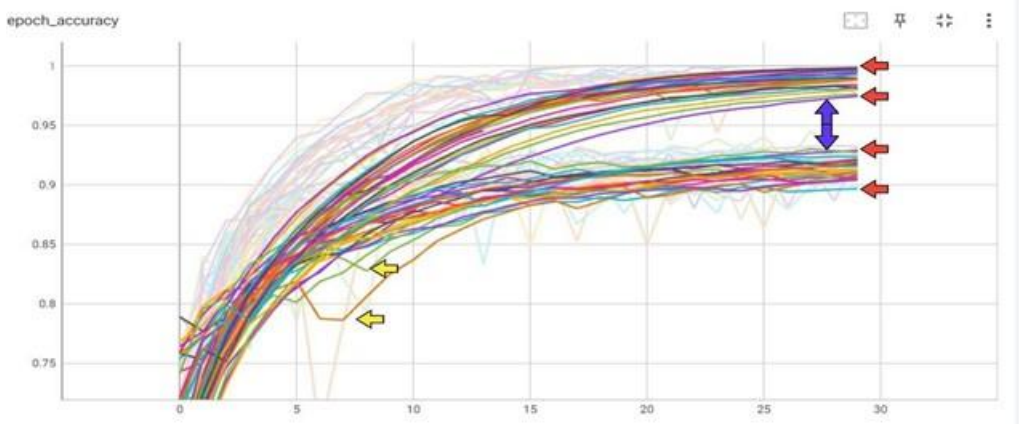
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 49, 24, 32)	832
max_pooling2d (MaxPooling2D)	(None, 24, 11, 32)	0
conv2d_1 (Conv2D)	(None, 24, 11, 32)	25632
max_pooling2d_1 (MaxPooling2D)	(None, 11, 5, 32)	0
flatten (Flatten)	(None, 1760)	0
dense (Dense)	(None, 128)	225408
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387

Total params: 252,259
Trainable params: 252,259
Non-trainable params: 0

Análisis de gráficas: La selección de modelos se fundamenta en diversos criterios cruciales. Al analizar la gráfica de "precisión" o "accuracy", nuestro objetivo fue observar un crecimiento constante y sin fluctuaciones abruptas tanto en la curva de precisión como en la de precisión de los datos de validación. La cercanía entre ambas curvas denota un rendimiento destacado del modelo, lo cual es un indicativo valioso. Es esencial asegurar que el modelo aprenda de manera efectiva en lugar de limitarse a memorizar datos.

Figura 16.

Precisión vs épocas de los 27 modelos entrenados.



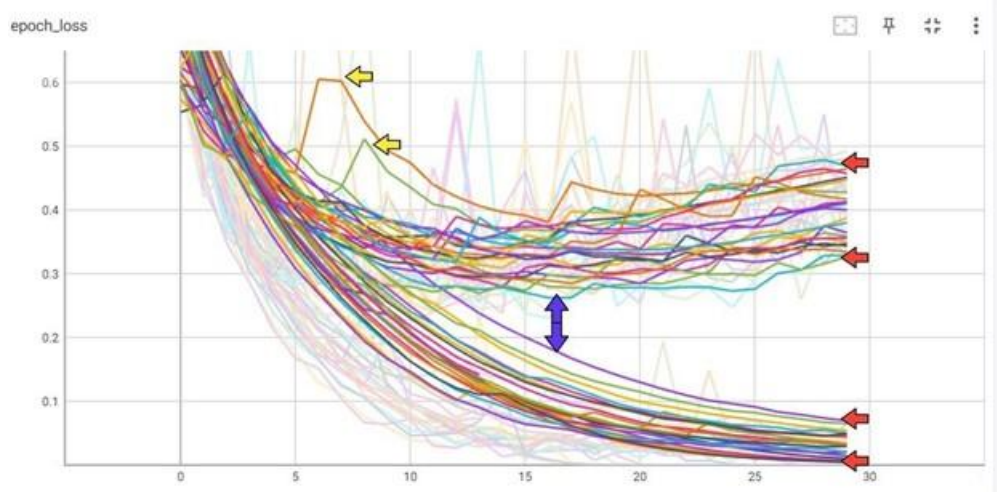
En esta representación gráfica, se aprecian los niveles de precisión ubicados en la parte superior, mientras que justo debajo se despliegan las curvas de precisión correspondientes a los datos de validación. Las flechas azules señalan la proximidad entre estos dos conjuntos de gráficos, evidenciando así el proceso de aprendizaje de la red. A través de la gráfica previa, es notable que la mayoría de los modelos exhiben un crecimiento constante, alineándose con la curva de precisión (con la excepción de aquellos indicados por las flechas amarillas).

Como criterio general, se destaca que a medida que el accuracy aumenta, el modelo demuestra un mejor entrenamiento. Este principio se refleja en las flechas de color rojo, donde los valores cercanos a '1' indican una mayor precisión.

Además, en la evaluación de modelos, es importante que la curva de pérdida en el conjunto de validación siga un patrón similar a la curva de pérdida general, evitando aumentos excesivos hacia valores elevados. Esto nos asegura que el modelo no esté sufriendo de un sobreajuste (overfitting) y que su capacidad de generalización sea adecuada.

Figura 17.

Pérdida vs épocas de los 27 modelos entrenados.



En el gráfico que representa las épocas frente a la pérdida, se evidencia que las curvas de pérdida tienden a converger hacia cero, mientras que la curva de pérdida en los datos de validación se mantiene en niveles más elevados. Es importante destacar que algunas curvas muestran valores más bajos que otras. Cuando el valor de pérdida es menor, existe una mayor probabilidad de que la red esté mejor entrenada. Las flechas amarillas señalan cambios abruptos en algunos modelos, indicando un entrenamiento deficiente y sugiriendo que esas redes deben ser descartadas. Además, se observa que los modelos más prometedores son aquellos señalados por las flechas rojas inferiores, ya que sus funciones de pérdida tienden a acercarse a cero, lo que indica un rendimiento óptimo.

Con base en estos valores se escogieron las 3 mejores redes que se destacaron en las 2 gráficas presentadas anteriormente, bautizándolas de la siguiente manera:

Red1: RedConv_F128_D128_dropout0.5 Red2: RedConv_F64_D128_dropout0.2 Red3: RedConv_F32_D256_dropout0.

Arquitectura AlexNet: Al obtener los resultados de la arquitectura anterior se optó por

buscar arquitecturas que tengan una validación a nivel investigativo respecto al área de clasificación multicategoría de imágenes. Es por ello que antes de evaluar los modelos con datos reales se contempló la siguiente arquitectura basada en AlexNet. Para la implementación de esta arquitectura se variaron los valores para determinar qué parámetros son óptimos para el problema que se desea resolver. En este caso se varió el número de capas densas y el número de filtros de las capas convolucionales con el fin de ver qué variación de la arquitectura AlexNet es más adecuada para este problema específico.

Tabla 4.

Arquitectura de las redes AlexNet.

Red	**Capa Conv. 1**	**Capa Conv. 2**	**Capa Conv. 3**	**Capa Conv. 4**	**Capa Conv. 5**	**Capa Densa 1**	**Capa Densa 2**
Alex 1	96	256	384	384	256	4096	
Alex 2	96	256	384	384	256	4096	4096
Alex 3	48	128	192	192	128	2048	
Alex 4	48	128	192	192	128	2048	2048

Nota: En la anterior tabla se puede ver la estructura de cada red basada en AlexNet, siendo los valores de “Capa Conv.” la cantidad de filtros convolucionales por capa y en los valores de “Capa Densa” la cantidad de neuronas por capa.

Los parámetros que no se variaron en las anteriores arquitecturas también se mantuvieron constantes en el entrenamiento de estos modelos.

El mejor desempeño fue obtenido por el modelo Alex2 y Alex3. Como se puede observar en las siguientes gráficas:

Figura 18.

Precisión vs épocas modelo Alex2

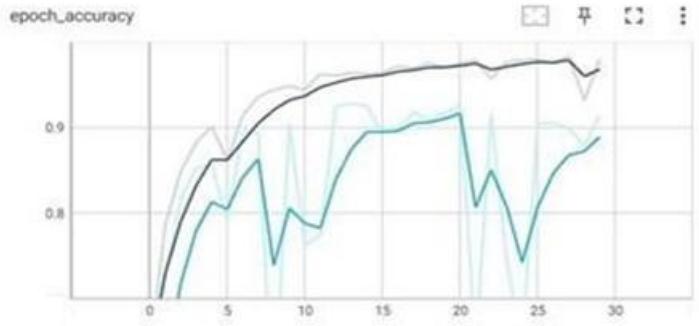
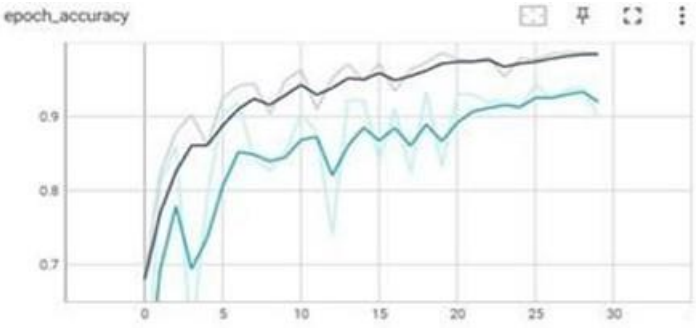


Figura 19.

Precisión vs épocas modelo Alex3



A pesar de que las gráficas nos dan un indicio de cómo se comporta la red neuronal a la hora de evaluar imágenes con vehículos. Se realizaron pruebas de estas redes con datos reales tomados directamente de la portería de la universidad.

Metodología de evaluación: Partiendo del primer filtro de selección del apartado anterior se procedió a seleccionar las 3 mejores redes de la arquitectura básica y las mejores 2 de la arquitectura AlexNet. Basándose en las muestras tomadas en la universidad en diferentes días y condiciones climáticas se ejecutó un código que genera una evaluación de precisión de cada uno de los modelos para poder evaluar su efectividad y funcionalidad operando en un entorno real y no con imágenes de entrenamiento.

Figura 20.

Cicla en la UIS

Figura 21.

Moto en la UIS

Figura 22.

Carro en la UIS

Figura 23.

Diagrama de la metodología usada.

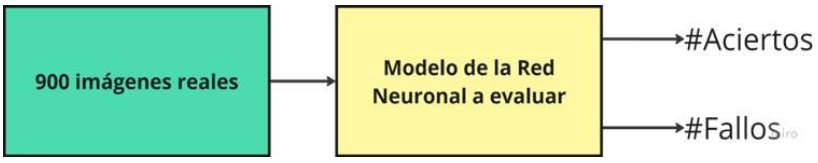


Tabla 5.

Comparativo de la precisión de las mejores redes.

Modelo	Aciertos	Fallos	Precisión Porcentual	Tiempo de Procesamiento Promedio
Red1	829	71	92.11%	26 ms
Red2	811	89	90.11%	23 ms
Red3	784	116	87.11%	25 ms
Alex2	492	408	54.66%	100 ms
Alex3	794	106	88.22%	85 ms

Nota: Los valores presentados en la tabla anterior reflejan tanto los aciertos como los fallos obtenidos por cada una de las redes durante una evaluación de 900 muestras de vehículos reales. También la tabla cuenta con los cálculos de la precisión porcentual y el tiempo de procesamiento de cada red.

Claramente el modelo que obtuvo resultados más desfavorables es el Alex2 con el menor porcentaje de acierto.

Los modelos Red1, Red2, Red3 y Alex3 tuvieron resultados muy similares con alrededor del 90% de precisión. En términos de tiempos de procesamiento no representan una diferencia muy importante, ya que no es imprescindible tanta velocidad de procesamiento para esta aplicación, aunque es evidente que los modelos Alex tienen un tiempo de procesamiento mayor a las redes básicas (son alrededor de 4 veces más lentas).

Evaluación por épocas: Para esta fase seleccionaron las 3 mejores redes para evaluar su rendimiento con valores de épocas más grandes. Dando así más posibilidad de que el entrenamiento llegue a un punto óptimo. Para ello se realizó una metodología similar a la anterior, pero teniendo en cuenta 3 valores de épocas diferentes para cada uno de los modelos. Generando

los siguientes resultados:

Tabla 6.

Evaluación de precisión variando la cantidad de épocas de entrenamiento.

Modelo	Aciertos	Fallos	Precisión Porcentual	Épocas
Red1	829	71	92.11%	30
Red1	784	116	87.11%	50
Red1	739	161	82.11%	80
Red 2	811	89	90.11%	30
Red 2	777	123	86.33%	50
Red 2	760	140	84.44%	80
Alex 3	794	106	88.22%	30
Alex 3	484	416	53.77%	50
Alex 3	424	476	47.11%	80

Nota: Esta tabla refleja el desempeño de cada red al ser sometida a variaciones en la cantidad de épocas de entrenamiento, mostrando la cantidad de aciertos y fallos obtenidos por cada una.

De los resultados consignados se puede concluir que el entrenamiento de un modelo de red neuronal convolucional no va a ser estrictamente mejor cuando se entrena por un mayor número de épocas. En este caso ocurrió lo contrario, a medida que las épocas aumentaban la precisión decaía.

La causa del fenómeno contemplado anteriormente se debe a un posible Overfitting, ya que la red está aprendiendo de memoria las imágenes de entrenamiento y pierde su capacidad de generalizar con datos reales captados en la universidad.

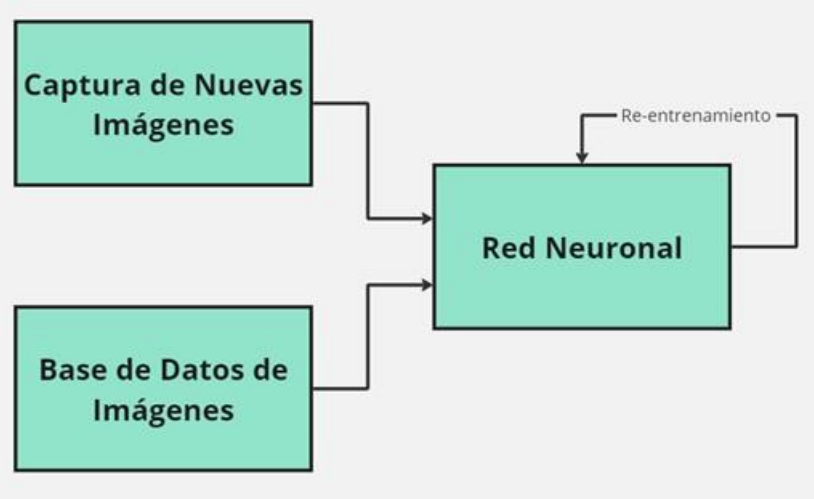
Debido a la eficacia que obtuvo la Red1, se seleccionó dicha red como la que presta mejores beneficios para el sistema.

Re-entrenamiento de la red neuronal: Con el fin de lograr un mejor ajuste del modelo de

red neuronal convolucional se optó por un sistema retroalimentado de aprendizaje automático supervisado. Para ello se propuso que las imágenes que se capturan para ser leídas por la red neuronal sirvan como retroalimentación para el sistema, afinando así su precisión a medida que se va usando.

Figura 24.

Diagrama de entrenamiento modelos re-entrenados.



Se considera un sistema de machine learning supervisado debido a que es indispensable que las nuevas imágenes que se estén ingresando a la base de datos tengan una validación humana. Esto con el fin de ingresar información totalmente verídica a la red neuronal. El encargado de verificar dicha información deberá clasificar manualmente entre las categorías de vehículos (carro, motocicleta, bicicleta) para que el código le asigne una etiqueta válida.

Una de las ventajas de implementar este sistema es que existe otro tipo de vehículos que ingresan al campus de manera menos frecuente pero que pueden estar estableciendo un sesgo en la precisión del modelo. Como lo son los vehículos de construcción, patinetas eléctricas, camiones y buses de diferentes tamaños, etc. Con esto se tendrá una mayor certeza de las predicciones que la red neuronal hará y se podría llegar a contemplar en añadir categorías adicionales para su

clasificación.

Metodología: Para verificar si el sistema realmente mejora su eficacia al ser realimentado con imágenes captadas en la universidad, se llevó a cabo un proceso de re-entrenamiento de la red utilizando 450 de las 900 imágenes que se utilizaron previamente para evaluar los modelos de la red neuronal. Este proceso se desarrolló siguiendo los siguientes pasos:

- Primero se clasificaron las imágenes tomadas en la universidad en las 3 categorías (carro, motocicleta, bicicleta)
- A continuación, se realizó aumento de datos con estas imágenes de la misma manera que se realizó con la base de datos inicial.
- Se entrenó de nuevo la red neuronal generando ahora un nuevo modelo retroalimentado.
- Luego se procedió a evaluar la red neuronal con los 450 vehículos restantes mezclados aleatoriamente para confirmar su efectividad. (No se usan las mismas 900 imágenes debido a que el sistema puede presentar overfitting y esto puede entregar un criterio erróneo.

Los resultados obtenidos fueron los siguientes:

Tabla 7.

Resultados de la red re-entrenada.

Modelo	Aciertos	Fallos	Precisión Porcentual	Tiempo de Procesamiento Promedio
Red re-entrenada	438	12	97.33%	31 ms
Red 1	415	35	92.22%	26 ms

Nota: En esta tabla se presenta una comparación del rendimiento entre la red convencional y la red re-entrenada, resaltando de esta manera las diferencias en cuanto a precisión y tiempo de procesamiento entre ambas.

De lo anterior se puede analizar que el modelo sí presentó una mejora de más del 5%. Esto puede deberse a que el sistema distingue mejor a los vehículos cuando están en el mismo entorno, sin presentar tantas variaciones en los diferentes ambientes de las imágenes originales del dataset. Asimismo, la posición de los autos y el tamaño puede influir en que la red aprenda mejor y establezca mejor los pesos entre las capas.

Figura 25.

Ejemplo de la imagen con la que fue originalmente entrenada la red comparada con la imagen usada en el re-entrenamiento.

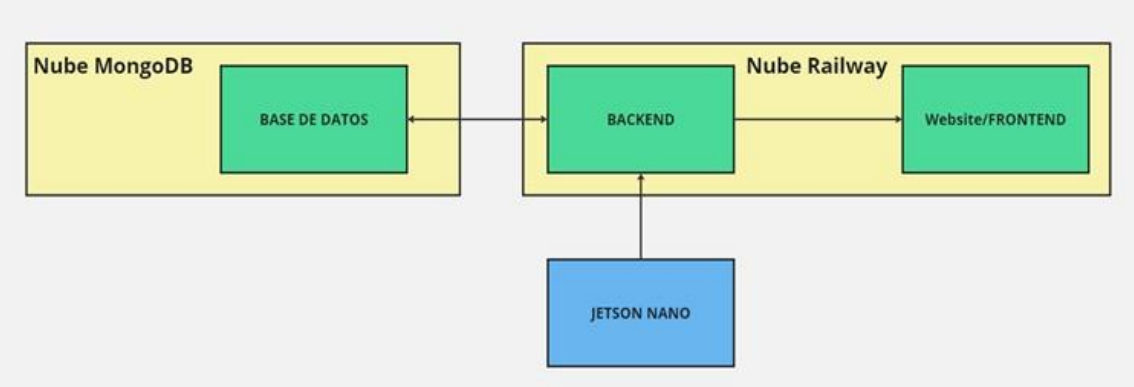
Si bien no se puede garantizar que el re-entrenamiento del modelo siempre conduzca a mejoras, este proceso indicó que un sistema realimentado tiene el potencial de ofrecer un rendimiento mejorado para esta aplicación. Sin embargo, surge un desafío evidente debido a la falta de practicidad, ya que implica la intervención humana, lo cual puede resultar laborioso al clasificar las imágenes capturadas por el sistema. Aunque esta es una característica intrínseca del aprendizaje supervisado de machine learning y es indispensable si se desea lograr un modelo más preciso.

5. Creación del Aplicativo

Para crear la interfaz HMI, se requiere la construcción de tres elementos fundamentales: la base de datos, el backend y el frontend. A grandes rasgos, la estructura de conexiones del diseño se describe en la siguiente figura:

Figura 26.

Interconexiones entre los módulos del HMI.



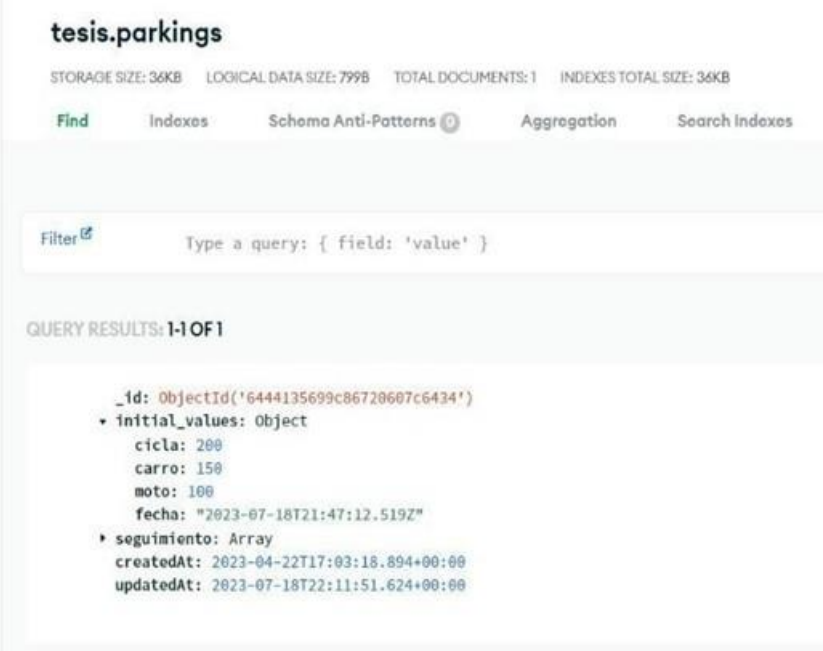
Base de Datos: Para la construcción de nuestra base de datos, se consideraron cuatro variables esenciales: automóviles, motocicletas, bicicletas y la fecha. Esto se hizo con el propósito de llevar un registro preciso de los vehículos que actualmente se encuentran en el campus universitario.

La base de datos se desarrolló utilizando Mongo.db, una base de datos NoSQL que se diferencia de las bases de datos relacionales tradicionales. Este tipo de sistemas está especialmente diseñado para administrar y almacenar grandes volúmenes de datos de manera eficiente y fiable.

Esta base de datos permitió mantener un histórico detallado de la cantidad de vehículos que ingresan y salen de la universidad, categorizados por tipo de vehículo y fecha, como se puede apreciar en la captura siguiente:

Figura 27.

Ejemplo de registro en la base de datos del proyecto.



La base de datos de Mongo.DB dispone de un sistema de almacenamiento en la nube que opera de manera independiente a la plataforma web principal. Esta configuración resulta esencial, ya que garantiza la seguridad de los datos incluso en situaciones en las que el almacenamiento interno de la Jetson Nano falle o sufra algún tipo de daño.

Backend: El Backend es el centro de la funcionalidad del HMI debido a que es el que realiza la interconexión entre la Base de Datos, la Jetson Nano y el Frontend. Este módulo se encarga de gestionar las peticiones y las promesas.

Para ello se utilizó Node.JS como entorno de ejecución de Javascript. El cual permite ejecutar líneas de código del lado del servidor de manera asíncrona, lo que permite realizar múltiples operaciones en tiempo real y para una afluencia alta. Estas operaciones se realizan en un servidor de RAILWAY, una plataforma que permite realizar el cómputo y alojar tanto el backend como el frontend. Para la gestión de las peticiones se han creado 5 servicios diferentes con el fin

de realizar actualizaciones y monitoreo de la información de la base de datos. Estos servicios son:

Tabla 8.

Servicios de backend.

Servicio	Url	Función
Consultar_todos	https://backend-fabian-production.up.railway.app/parking/consultar	Este servicio arroja toda la información que está contenida en la base de datos.
Crear_parking	https://backend-fabian-production.up.railway.app/parking/crearParking/50/100/200	Sirve como un paso de inicialización para crear el registro y así permitir que se añadan datos.
Reset_parking	https://backend-fabian-production.up.railway.app/parking/resetParking/50/100/200	Es muy útil cuando se está realizando pruebas o se desea reiniciar los valores de "ingreso" y "salida" a cero. Además, este servicio permite establecer la cantidad de parqueaderos iniciales.
Sumar_parking	https://backend-fabian-production.up.railway.app/parking/sumar/carro	Este servicio pide como requisito el "Tipo de vehículo" y sumará un vehículo cuando ingrese al campus.
Restar_parking	https://backend-fabian-production.up.railway.app/parking/restar/carro	Restar_parking se encarga de restar un vehículo cuando sale del campus, indicando el tipo de vehículo correspondiente.

Nota: Esta tabla cuenta con la descripción y los URL de todos los servicios que se usa en el backend para el funcionamiento del sistema.

Servidor: Para que la página web esté disponible y sea accesible desde cualquier dispositivo a través de Internet, es necesario alojar tanto el Backend como el Frontend en un servidor. Este servidor actúa como el hosting del proyecto, permitiendo que se almacene en la nube y facilite la correlación de variables a través de solicitudes y puntos de acceso ("endpoints"). Además de servir como espacio para ejecutar el Backend sin la necesidad de depender de un servidor local (una computadora), este enfoque proporciona seguridad al garantizar la estabilidad y el funcionamiento constante del sistema durante largos periodos de tiempo. Para acceder a la página web se ingresa a la siguiente dirección: https://frontend-fabian-098f22.netlify.app/. La página luce de la siguiente manera:

Figura 28.

Página web del proyecto.

Como se puede observar la página cuenta con 8 campos que se actualizan desde la base de datos. La información que proporciona el sistema es muy intuitiva y fácil de leer. Está pensada para que sea lo más entendible posible.

La interfaz se divide en 3 apartados, “Ingresos”, “Salidas” y “Espacios Disponibles”.

El módulo de ingresos realizará la cuenta de la entrada de vehículos al campus discriminado

por tipo de vehículo. De la misma manera el de salidas registrará toda salida que se efectúe el día en curso. Claramente estos dos apartados no tendrán relación el uno con el otro ni tampoco con la capacidad de vehículos que la universidad puede albergar.

El tercer módulo es el más útil ya que es el que entrega información en tiempo real de los estacionamientos disponibles para cada tipo de vehículo. Hay que tener en consideración que los estacionamientos de bicicletas no se pueden cuantificar, ya que es un vehículo bastante pequeño y no cuenta con parqueaderos limitados dentro del campus de la universidad. Además, se debe inicializar los valores de parqueaderos disponibles para que la base de datos reconozca cuantos espacios de parqueo existen actualmente en la universidad.

Esta información se actualizará cada 5 segundos con el fin de que la persona que esté usando el sistema no tenga que actualizar la página manualmente y pueda acceder a estos datos.

A modo de ejemplo cuando la tarjeta detecta que un vehículo está atravesando el rango visual de la cámara tomará una captura y lo analizará para determinar tipo de vehículo y sentido. En este caso detectó una motocicleta entrando al campus.

Como valores iniciales se tiene: 150 parqueaderos disponibles de carro y 90 de motocicleta. El código de Upload enviará al servidor la siguiente petición: https://backend-fabian-production.up.railway.app/parking/sumar/moto

Figura 29.

Ejemplo de cómo se actualizan los valores en la página web.

Como se puede observar en el módulo de Ingresos se realizó un aumento en la variable de “Motos”. Por otro lado, en el apartado de “Espacios Disponibles” se realiza una resta, ya que de los 90 parqueaderos disponibles de moto se disminuye en ‘1’ dicho valor.

6. Pruebas en el Campus

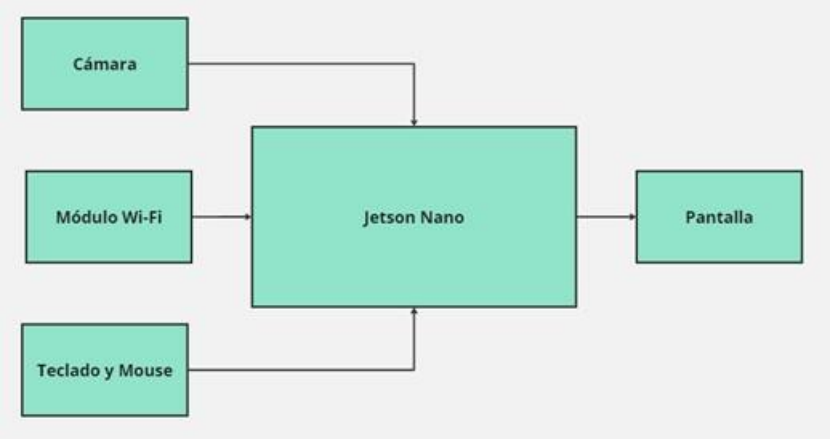
Hardware: El sistema se compone de 2 partes, el hardware y el software. El hardware consta de 3 entradas al sistema, la cámara como entrada principal, encargada de capturar la información exterior y emitirla a la tarjeta Jetson Nano. Además, se tiene 2 entradas periféricas que no están involucradas directamente con el proceso, pero son herramientas necesarias para el monitoreo en fase de prueba.

El módulo Wi-Fi es necesario para establecer la conexión con la base de datos que estará almacenada en la nube. Esto con el fin de proteger los datos y no depender tan sólo de los datos locales que se podrían almacenar en la Jetson Nano.

El Teclado y el mouse se usan para controlar la sesión en la tarjeta y corregir cualquier eventualidad que pueda ocurrir en el tiempo de operación. De la misma manera se tiene la pantalla para poder visualizar la página local en la cual estará actualizándose la información actual de vehículos.

Figura 30.

Diagrama de interconexión de los periféricos que usa la Jetson Nano.



Software

Etapa de visión artificial: El sistema se basa principalmente en la utilización de imágenes captadas por una cámara ubicada en el lugar de interés. Estas imágenes deben someterse a un proceso de visión artificial para extraer las características relevantes necesarias para su posterior clasificación.

Para lograr una clasificación precisa, es esencial asegurar que las capturas se realicen de manera adecuada, evitando así cualquier sesgo o almacenamiento innecesario de datos. Este proceso implica la modificación de la dimensión, los colores y el fondo de la imagen con el fin de identificar los objetos en movimiento presentes en ella. Posteriormente, se realiza un seguimiento de estos objetos, extrayendo sus características físicas, como el ancho, alto, centro y dirección. Utilizando estos parámetros, se procede a eliminar los contornos no deseados y a almacenar únicamente los elementos de interés mediante una nueva captura.

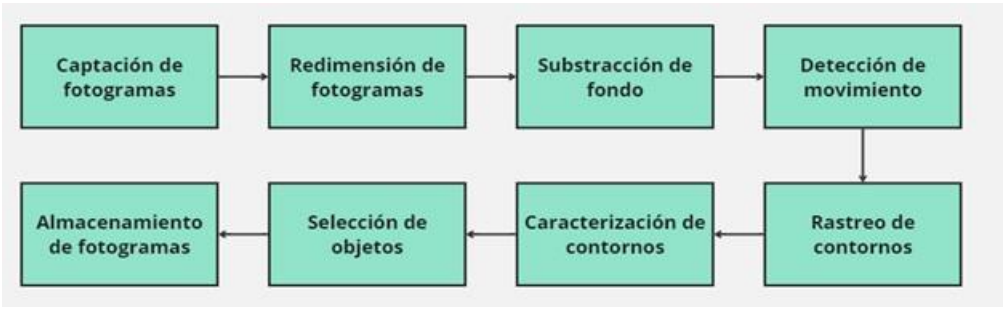
Figura 31.

Motocicleta siendo captada por el sistema

Nota: la imagen full color(izq) versus su máscara a blanco y negro (der).

Figura 32.

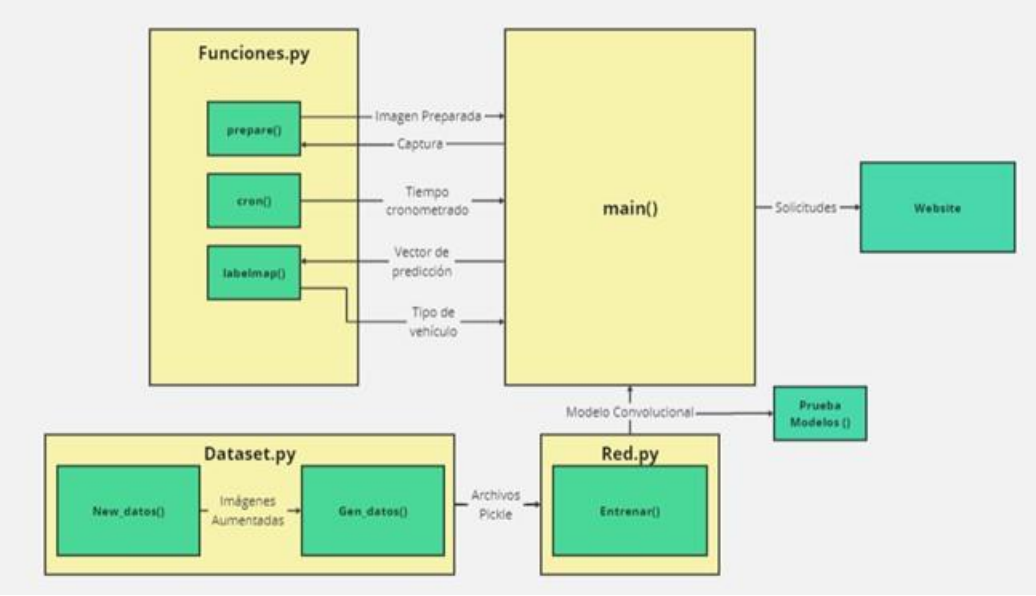
Paso a paso del tratamiento de las imágenes con visión artificial.



En principio, es crucial que la cámara tenga un campo de visión despejado, sin interferencias naturales o bloqueos causados por errores humanos. Estas obstrucciones pueden distorsionar el conteo de vehículos o la identificación de su tipo. No obstante, el proceso de visión artificial se encarga de separar los elementos de interés de tales obstáculos, reduciendo así el riesgo de que puedan afectar las detecciones. El siguiente diagrama conjuga todas las interconexiones entre las funciones y el diseño del software que hace parte del sistema.

Figura 33.

Diagrama de bloques del funcionamiento del software del sistema.



Funciones.py: contiene todas las funciones que no representan más de 20 líneas de código. Como lo son labelmap(), prepare() y cron().

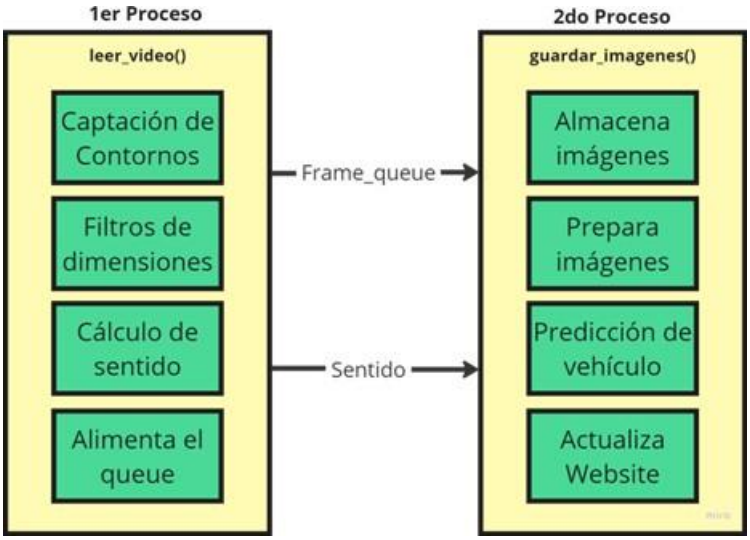
Red.py: cuenta con la función entrenar(), la cual se encarga de entrenar el modelo y generar el archivo .H5, el cual podrá ser cargado para realizar predicciones en otros códigos.

Dataset.py: contiene la función New_datos() y Gen_datos(). Las cuales se encargan de leer las imágenes guardadas en la carpeta del proyecto y les asigna un label para reconocer qué tipo de vehículo es.

En función de diseñar un sistema más eficiente se optó por utilizar un sistema multiprocesos. Esto con el fin de tener 2 procesos que se efectuan en simultáneo de manera asíncrona, liberando así la carga computacional de la tarjeta. Estos procesos están contenidos dentro de la función main(), complementando así su funcionamiento en conjunto con otras funciones. Esta función es la encargada de integrar todas las funcionalidades del sistema como se puede ver en el siguiente esquema:

Figura 34.

Diagrama del funcionamiento del sistema dividido en multiprocesos.



A continuación, se describe de manera breve el funcionamiento de cada una de las funciones utilizadas en el sistema.

Gen_datos(): es la función encargada de tomar las imágenes del dataset y convertirlas en un formato que pueda ser leído por la red neuronal para su posterior entrenamiento.

New_datos(): esta función es usada para realizar aumento de datos o “data augmentation”. Toma el dataset de imágenes, realiza los cambios basados en parámetros de entrada (posición, zoom, reflexión, etc) y posteriormente las almacena en otra carpeta para que se actualice y de ahí se generen los archivos Pickle.

Entrenar(): es la encargada de realizar el entrenamiento de la red neuronal. En esta se describe la arquitectura y todos los parámetros relevantes para el entrenamiento. Desde los tipos de funciones de activación hasta las épocas de entrenamiento. Esta función carga los archivos Pickle y los usa para el aprendizaje de la red.

Prueba Modelos: es un archivo de python diseñado con el fin de evaluar los modelos que se desea poner a prueba con una base de datos de imágenes de prueba. En este caso específico se

cargan las imágenes que se capturaron en la Universidad Industrial de Santander, con el fin de evaluar las redes con datos reales. Esta función calcula la cantidad de aciertos que tiene el modelo con respecto a la cantidad de imágenes que existen en total en el directorio en cuestión.

Prepare(): esta función transforma las imágenes para que puedan ser evaluadas por la red neuronal. A cada dato lo redimensiona y le elimina los canales innecesarios para la posterior clasificación.

Labelmap(): esta función transforma los valores que entrega el modelo al leer una imágen y los interpreta para clasificarlo en la categoría correspondiente, entregando el valor de “carro”, “moto” o “cicla”. Además, realiza validaciones para evitar errores, ya que en ciertas ocasiones la red puede entregar valores que no son válidos y deben ser corregidos.

Las anteriores funciones están contenidas en los archivos de Python mencionados anteriormente, los cuales pueden ser consultados en el repositorio del proyecto.

El bloque “Website” hace referencia a la estructura computacional que se ha descrito en el diseño del HMI.

Metodología de Pruebas: El sistema embebido se instaló en la entrada de la cra 25 de la Universidad Industrial de Santander para obtener los datos de ingreso y salida de vehículos con el fin de poner a prueba el funcionamiento del sistema. El funcionamiento del sistema se puede ver en los enlaces que se encuentran en el README del repositorio, en unos fragmentos de video extraídos del día de la prueba.

Esta actividad se realizó un total de 6 horas y el montaje del sistema se realizó de la siguiente manera:

Figura 35.

Sistema embebido compuesto de la Jetson Nano y sus periféricos.

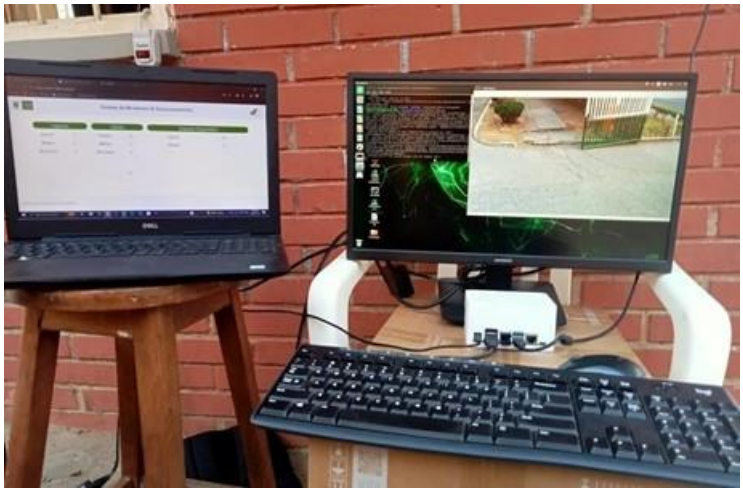


Figura 36.

Tarjeta Jetson Nano (Carcasa blanca) operando.

Es imprescindible aclarar que el sistema depende de los periféricos solo para su monitoreo e inicialización del programa en la tarjeta. Para el funcionamiento adecuado del sistema solo es necesario conectar la cámara web para captar las imágenes.

Para el correcto conteo de espacios se tuvo en cuenta los valores iniciales de vehículos dentro del campus con el fin de conocer los espacios disponibles al momento de empezar las

pruebas. Los valores iniciales de cantidad de vehículos son: carros=45 y motos= 3, tendiendo así los siguientes resultados:

Tabla 9.

Resultados del sistema operando en tiempo real.

	Carros I	**Carros O**	**Motos I**	**Motos O**	**Ciclas I**	**Ciclas O**
Sistema	83	66	17	16	16	13
Real	85	70	16	16	18	14
Error %	2.35%	5.71%	5.88%	0.00%	11.11%	7.14%

Nota: Esta tabla ilustra una evaluación comparativa que abarca tanto el rendimiento global del sistema como los datos reales tomados de manera manual. La I representa la entrada y la O la salida de cada tipo de vehículo.

Las pruebas tuvieron como resultado una eficacia del sistema de 96.34%.

De los anteriores resultados se puede observar que se tiene un error en general del 3.66%. Hay que tener en cuenta que cada una de las fases del proyecto agrega un porcentaje de error, desde la captura en la fase de visión artificial, hasta la clasificación en la red neuronal convolucional.

Existen 2 factores principales puede influir y podrían bajar la precisión del sistema. Por un lado, el sistema está entrenado con vehículos tradicionales (automóviles, motos y bicicletas), pero cuando la cámara capta vehículos como motocarros, patinetas eléctricas, carretillas o vehículos de construcción puede llegar a interpretarlos de manera incorrecta. Para esto es importante realizar el re-entrenamiento de la red y así tener una base de datos de entrenamiento con más variedad de vehículos.

Además, el otro factor que influye es el clima, ya que un sol demasiado fuerte o una alta temperatura ambiental puede hacer que la tarjeta se recaliente y no procese bien la fase de visión

artificial. Para esto se recomienda un sistema de refrigeración que evite este inconveniente.

Al final del ejercicio se obtuvieron los resultados mostrados en la website de la siguiente manera:

Figura 37.

Website al final de las pruebas.

7. Conclusiones

Debido a la eficacia y el tiempo de procesamiento de la red, se concluye que la Red1 (RedConv_F128_D128_dropout0.5) con el reentrenamiento es la opción más idónea para esta aplicación específica.

El modelo experimenta mejoras notables tras someterlo a un proceso de reentrenamiento utilizando la base de datos de imágenes recopilada durante el funcionamiento del sistema, Se recomienda realizar re-entrenamientos regularmente para elevar la efectividad del sistema.

El aplicativo web desarrollado es funcional y eficiente, dando así la información necesaria para conocer el estado de la capacidad de los parqueaderos dentro del campus universitario.

Con el fin de evitar que la tarjeta presente fallos en la fase de visión artificial es indispensable implementar un sistema de refrigeración para evitar el recalentamiento de esta.

Para lograr un sistema más robusto y preciso, se sugiere incorporar en el entrenamiento del modelo todo tipo de vehículos que accedan al campus.

Referencias

Alarcón, Carlos. (2023). Curso de Redes Neuronales Convolucionales con Python y Keras. *Platzi*. https://platzi.com/cursos/redes-neuronales-convolucionales/

Bangar, Siddhesh. (2022). *AlexNet Architecture Explained*. https://medium.com/@siddheshb008/alexnet-architecture-explained-b6240c528bd5

Basogain Olabe, X. *Redes Neuronales Artificiales y sus Aplicaciones.* https://ocw.ehu.eus/pluginfile.php/40137/mod_resource/content/1/redes_neuro/contenidos/pdf/libro-del-curso.pdf

Caicedo, E.F., & López, J.A. (2009). Una aproximación práctica a las redes neuronales artificiales. *Revista en Cursiva*. Editorial: Programa Editorial de la Universidad del Valle.

Henderson, H. (2009). Computer Science and Technology. *Facts On File.*

Matich, D.J. (2001). *Redes Neuronales: Conceptos Básicos y Aplicaciones*. Tesis presentada a la Universidad Tecnológica Nacional – Grupo de Investigación Aplicada a la Ingeniería Química (GIAIQ).

McKellar, J. (2021). *Parking Guidance Systems*. https://blog.nortechcontrol.com/parking-guidance-systems

MongoDB. (2023). *Guías de MongoDB.* https://www.mongodb.com/docs/guides/

NVIDIA. (s.f.). *Kit de Desarrollo NVIDIA Jetson Nano.* https://developer.nvidia.com/embedded/jetson-nano-developer-kit# OpenCV. (s.f.). *Acerca de OpenCV.* https://opencv.org/about/

Secretaría de Tránsito de Bucaramanga. (2015). *Parque Automotor año 2015*. [Formato de archivoPDF]

Secretaría de Tránsito de Bucaramanga. (2018). *Jornada de la Movilidad Sostenible y Promoción de Modos de Transporte Sostenible*. [Formato de archivo PDF].

Secretaría de Tránsito de Bucaramanga. (2022). *Parque Automotor año 2022*. [Formato de archivo PDF]

Solem, J. E. (2012). *Programming Computer Vision with Python. O'Reilly Media*. Licencia Creative Commons [CC BY-NC-ND 3.0].

TensorFlow. (2023). *Guía para comenzar con TensorBoard*. https://www.tensorflow.org/tensorboard/get_started?hl=es-419

TensorFlow. (s.f.). *Descripción general de TensorFlow*. https://www.tensorflow.org/overview?hl=es

TensorFlow. (s.f.). *Tutorial de Aumento de Datos en Imágenes con TensorFlow*. https://www.tensorflow.org/tutorials/images/data_augmentation

Universidad Industrial de Santander. (2019). *Acuerdo No. 047 de 2019*. Bucaramanga, Colombia.