DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 14 No. 27 (2026) 206-214

# Red neuronal convolucional para control de activación y paro de un dispositivo electrónico
# Convolutional neural network for controlling the activation and stop of an electronic device

Mario Alexis Ibarra-Yáñez[a], Kevin Uriel Chávez-Castro[a], Mauricio Jesús Labastida-Cuevas[a], David Tinoco-Varela[b, *], José de Jesús Morales-Romero[b]

[a] *Ingeniería en telecomunicaciones, sistemas y electrónica. Facultad de Estudios Superiores Cuautitlán.*
[b] *Departamento de ingeniería. Facultad de Estudios Superiores Cuautitlán.*

**Resumen**

Hoy en día, los sistemas de visión artificial han tomado gran relevancia en diversos tipos de sistemas, realizando tareas tales como detección de rostros, detección de objetos, mapeo, etc. Estos sistemas descansan sobre diferentes algoritmos y herramientas computacionales para su funcionamiento, tales como las redes neuronales convolucionales. En este artículo se presenta un sistema de visión artificial que ha sido entrenado para poder activar o desactivar un robot móvil mediante el uso de una red neuronal convolucional (CNN). Tiene la característica de que realiza un control inalámbrico por medio de Wi-Fi y la tarjeta ESP32-CAM. Uno de los principales aportes de este trabajo es que la CNN ha sido entrenada con una base de datos de creación propia y que se puede considerar pequeña, con la finalidad de identificar tres clases de salida (o de control): “pelota”, “stop” y “nada”. El funcionamiento general del sistema consiste en la captura de imágenes del entorno por medio de la ESP32-CAM. Los datos capturados se transmiten vía Wi-Fi a una estación de cómputo con MATLAB para el procesamiento mediante una CNN, posteriormente MATLAB envía la respuesta y la clasificación resultante nuevamente a la ESP32-CAM. En función de la respuesta, el dispositivo electrónico móvil es activado, desactivado o permanece inmóvil. Adicionalmente, se creó una interfaz gráfica que muestra los datos esenciales relacionados a la captura de imagen y la respuesta de la red neuronal. El sistema tuvo una eficacia entre el 85% y el 90% en su funcionamiento, lo que se considera una respuesta adecuada a un entrenamiento con una base de datos pequeña.

*Palabras Clave: Visión artificial, Redes neuronales convolucionales, Control inalámbrico*.

**Abstract**

Nowadays, artificial vision systems have grown in relevance in many kinds of systems, performing different types of tasks, such as facial recognition, object detection, mapping, etc. Such systems are based on different types of algorithms and computational tools in order to reach their objectives, algorithms such as a Convolutional Neural Network are widely used. This paper presents an artificial vision system which has been trained to activate or deactivate an electronic device, all of this through the use of a Convolutional neural network (CNN). It can make wireless control by Wi-Fi and the ESP32-CAM board. One of the main contributions of this work is that the CNN has been trained using a database that can be considered small and self-created, with the purpose of identifying 3 output classes: “ball”, “stop” and "nada". The general behavior of the system is given by: capturing images of the environment using the ESP32-CAM; transmitting the images to a computer with MATLAB on it, in which the CNN is processed; and the subsequent transmission by MATLAB and the reception of the resulting classification by the ESP32-CAM; depending on the response, an electronic device is activated, deactivated, or remains inactive. Additionally, a graphic interface was created, this interface shows the essential data related to image capture and the neural network’s response. The system achieved an efficiency between 85% and 90% in its operation, which is considered an appropriate response to training with a small database.

*Keywords: Artificial vision, Convolutional neural networks, Wireless control*.

*Autor para la correspondencia: dativa19@hotmail.com
**Correo electrónico**: mario24.iy@gmail.com (Mario Alexis Ibarra Yáñez), kevinchavez1622@gmail.com (Kevin Uriel Chávez Castro), mauriciolabastida19@gmail.com (Mauricio Jesús Labastida Cuevas), dativa19@hotmail.com (David Tinoco Varela), jesusmoralesro@gmail.com (José de Jesús Morales Romero)

**Historial del manuscrito:** recibido el 16/07/2025, última versión-revisada recibida el 05/02/2026, aceptado el 19/02/2026, en línea (postprint) desde el 11/03/2026, publicado el 05/07/2026. **DOI:** https://doi.org/10.29057/icbi.v14i27.15537

## 1. Introducción

El uso de diversos sistemas de Inteligencia Artificial (IA) en las aplicaciones modernas está cada vez más presente. La IA se ha posicionado en prácticamente todos los espacios de la vida cotidiana, académica y social, entre otros ámbitos.

Existen diferentes algoritmos de IA que permiten procesar, de diversas formas, los datos provenientes del mundo físico (por ejemplo, los adquiridos por sensores) o provenientes de distintas bases de datos. Dentro de estos algoritmos están aquellos que nos permiten interactuar con el mundo por medio de visión artificial.

La visión artificial es una herramienta que se utiliza hoy en día en diferentes aplicaciones, como detección de rostros (Coşkun, et al., 2017), identificación de características específicas como, por ejemplo, en una fruta (Naranjo-Torres et al., 2020), también en sistemas de vuelo (Amer et al., 2021), sistemas robotizados (Williams et al., 2019) y, por supuesto, en sistemas de control autónomo (Rausch et al., 2017), por mencionar algunos.

En sistemas de control autónomo de robots, el componente crítico para una máquina es la capacidad de interpretar visualmente su entorno. Proyectos orientados a la creación de vehículos de exploración autónoma (Lu et al., 2019) o sistemas de asistencia a la conducción (Sakhare et al., 2020), por mencionar algunos, dependen del análisis de imágenes en tiempo real para una toma de decisiones adecuada. En este contexto, las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) han funcionado como una tecnología adecuada dada su notable eficacia en el reconocimiento de patrones complejos dentro de datos visuales.

Las CNN están diseñadas para procesar, principalmente, imágenes y son capaces de extraer características de las imágenes. Algunas de las características que se pueden extraer son detección de bordes, de texturas, de sombras, así como formas y objetos complejos. Dichas características permiten un reconocimiento de objetos robusto y confiable.

El proyecto que se presenta en este artículo tiene como finalidad generar y entrenar una CNN para que pueda identificar tres clases y, con estas clases, logre controlar el encendido y apagado de un dispositivo electrónico o robot móvil.

El robot móvil a controlar, con ayuda de una tarjeta de desarrollo ESP32-CAM, capturará imágenes de su entorno y estas serán analizadas por una CNN que se ejecutará en MATLAB. Dicha CNN está entrenada para la detección de tres clases de objetos. Las tres clases consideradas son: “Pelota” que es un objeto que se utilizó para indicarle al sistema que puede iniciar su funcionamiento, se ha de mencionar que este objeto fue elegido por simplicidad de aprendizaje debido a que la base de datos es realizada de forma manual; "Stop" que es la palabra que le indicará a nuestro sistema que debe de detener su funcionamiento, esta imagen fue utilizada debido a que es una representación visual que ya tiene de forma explícita la orden de detenerse; Por último, la clase “Nada”, la que nos indica que no existe el estímulo de activar el sistema o, en caso de estar activo, detener el sistema.

El objetivo principal es lograr que el vehículo tenga respuesta autónoma ante la detección de estos objetos, logrando que la CNN reaccione como un sistema de control en tiempo real, simulando así funcionalidades básicas en robots exploradores o vehículos con cierto nivel de autonomía.

## 2. Conocimientos previos

### *2.1. Redes Neuronales Convolucionales (CNN)*

Una de las áreas más conocidas dentro de la IA es aquella que se enfoca en el aprendizaje por medio de Redes Neuronales Artificiales (ANN). Existen diferentes tipos de arquitecturas de redes neuronales que van desde una neurona simple, tal como un perceptrón, hasta arquitecturas complejas que se enfocan en resolver diferentes problemáticas, tales como: reconstrucción de información, identificación de patrones dentro de un conjunto de datos, reconocimiento y procesamiento de texto y procesamiento de imágenes. Esta última tarea es de interés para los fines de este proyecto de investigación. Las CNN, además de contener la capa de convolución, pueden contener capas adicionales. A continuación, se describen los componentes principales de una CNN:

a. Capa Convolucional (*Convolutional Layer*): Es la capa de entrada y el núcleo de una red neuronal convolucional. En esta capa, se aplican filtros (también llamados *kernels*) a la imagen de entrada. Un filtro es una pequeña matriz de pesos y sesgos (*bias*) que se aplica sobre la imagen, realizando una operación de convolución. La capa convolucional utiliza una función de activación, del mismo modo que el resto de la ANN. Aunque se pueden utilizar diferentes funciones, generalmente es utilizada la función *Unidad Lineal Rectificada* (también conocida como ReLU). Esta función tiene la característica de que introduce no linealidades, lo que hace que la red se pueda ajustar a comportamientos más complejos. La operación de convolución junto con la función de activación ayuda a extraer características de la imagen. Algunas de las características pueden ser detección de bordes, esquinas y texturas, entre otras.
b. Capas convolucionales adicionales: Una capa convolucional adicional se coloca a la salida de la primera capa, y a la salida de esta puede colocarse otra capa y así sucesivamente. Colocar diversas capas adicionales permite crear jerarquías de características, esto ayuda a que la red neuronal pueda interpretar y extraer patrones relevantes. En la Figura 1 se observa un ejemplo básico del funcionamiento de una capa convolucional. Primero se secciona una parte de la imagen la cual es procesada a través de un filtro, después de esta operación, se realiza una función ReLU sobre la matriz resultante, dando como resultado la salida, que es la suma de los valores 12, 100 y 3.
c. Capas de Agrupación (*Pooling Layers*): Las capas de agrupación permiten mantener las características más relevantes de la imagen, asimismo, estas capas sirven para reducir la dimensionalidad de las matrices. Esto se traduce en una reducción en los procesos de cálculo. Una de las funciones más utilizadas es el *MaxPooling*, que selecciona el valor más grande dentro de una ventana.
d. Capas Completamente Conectadas (*Fully Connected Layer*s): Una vez que ha sido realizado el proceso convolucional y de *pooling*, los mapas de características pasan por un proceso de aplanado, es decir, convierten

matrices de dos dimensiones en vectores de una dimensión. Estos vectores entran en la capa completamente conectada, la cual realiza la clasificación de los datos obtenidos, mandando esta información a todas las neuronas de salida.

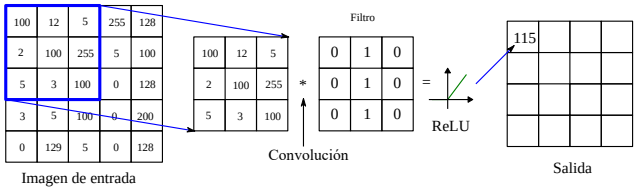


Figura 1. Representación básica de una CNN.

Como se mencionó anteriormente, las CNN han funcionado adecuadamente en el diseño de diversos vehículos autónomos. De acuerdo con Sainath *et al.* (2021), los vehículos autónomos se han desarrollado rápidamente en la última década gracias a los avances en aprendizaje profundo. Existen varios ejemplos de este tipo de dispositivos en la literatura científica, por ejemplo, Sadeghi *et al.* (2022) diseñaron un robot terrestre llamado SROBO que identifica su posición y es capaz de navegar por ciertas áreas utilizando una red neuronal convolucional profunda y aprendizaje por transferencia. En la misma línea, Sanango (2022) desarrolló un sistema de visión artificial basado en redes neuronales convolucionales para la detección de señales de tránsito, control de velocidad y detención, implementado sobre un vehículo autónomo.

Por otro lado, Bojarski *et al.* (2016) desarrollaron una CNN que genera comandos de control de dirección; según los autores, el sistema aprendió a conducir en el tráfico de carreteras locales con o sin marcas de carril y en autopistas, así como en zonas con guía visual poco clara, como estacionamientos y caminos sin pavimentar.

Shoeb *et al.* (2022) realizaron un prototipo de coche autónomo con visión artificial utilizando OpenCV2 y aprendizaje automático, el vehículo detecta el trazado del carril, las señales de tráfico y los semáforos, y responde al tráfico en tiempo real. Ellos utilizaron la tarjeta Raspberry Pi como unidad central de procesamiento.

Por su parte, Jain (2018), propone un modelo de coche autónomo capaz de desplazarse de un lugar a otro funcionando en diferentes tipos de pistas tales como curvas, rectas y rectas seguidas de curvas. Él montó un módulo de cámara sobre el coche y una Raspberry Pi, además, utilizó una CNN para predecir una de las siguientes direcciones: derecha, izquierda, avance o detención. En la misma línea, Seth *et al.* (2020) también realizaron la implementación de un vehículo autónomo basado en una red neuronal convolucional, utilizando una placa Raspberry Pi 4.

Otro diseño, propuesto por Almusawi *et al.* (2022), es un prototipo de coche totalmente autónomo que emplea visión artificial para detectar carriles y señales de tráfico. El proyecto utiliza una Raspberry Pi 3, que actúa como unidad de procesamiento de imágenes y aprendizaje automático y un sistema de visión artificial que corresponde a la biblioteca OpenCV2. Según los autores, el sistema fue capaz de detectar el carril y responder a los cambios de dirección, así como de detectar señales de tráfico y proporcionar las respuestas adecuadas.

Como es posible notar, existen diferentes tipos de propuestas relacionadas con vehículos autónomos, utilizando sistemas de visión artificial. En este caso, estamos agregando la comunicación inalámbrica y nuestro sistema visual de interpretación de datos. La comunicación inalámbrica nos permite tener un control sin interferencias al momento de que el vehículo "toma sus decisiones" de acción. Además, nos permite liberar recursos para el procesamiento de los datos dentro del dispositivo móvil, dejando todo el peso del procesamiento a un sistema que puede estar en un servidor local o en la nube.

### *2.2. ESP32-CAM*

La ESP32-CAM (Figura *2*) es una placa de desarrollo compacta y de bajo costo que combina un microcontrolador ESP32-S con una cámara. Sus características principales la hacen ideal para proyectos de visión artificial e Internet de las Cosas (IoT, por sus siglas en inglés) ya que cuenta con conectividad Wi-Fi y Bluetooth integrada; pines GPIO para realizar conexiones de sensores y actuadores; y es complementada con una cámara que permite capturar imágenes y vídeo.

Es importante mencionar que esta tarjeta ha sido utilizada en una gran variedad de proyectos tecnológicos tales como sensores de temperatura utilizando tecnología IoT (Rusimamto *et al.*, 2021), sistemas de seguridad (Cahyono *et al.*, 2022), monitoreo del crecimiento de plantas en sectores agrícolas (Elhattab *et al.*, 2023), el control de una cerradura inteligente utilizando tecnología IoT (Prathapagiri & Kosalendra, 2021), control de accesos mediante IoT (Raju et al., 2022), entre muchas otras aplicaciones.

Se observa que es ampliamente utilizada en esquemas dentro del IoT.

En este proyecto, la ESP32-CAM se utiliza para capturar el entorno del robot por medio de visión artificial, actuar como un transmisor de imágenes vía Wi-Fi y controlar los motores del robot según las instrucciones recibidas, instrucciones obtenidas después del análisis de imágenes realizado por MATLAB.


Figura 2. Placa de desarrollo ESP32-CAM.

## 3. Metodología y análisis

Para llevar a cabo este proyecto, la metodología se centró primero en la recolección y organización de los datos necesarios para el entrenamiento de la red neuronal, para luego pasar al diseño y programación del sistema completo.


Figura 3. Arquitectura dada en bloques de la red neuronal convolucional utilizada en el desarrollo del presente Proyecto.

### *3.1. Preparación del conjunto de datos*

Para conformar la base de datos que se utiliza en este proyecto, se tomaron directamente las fotografías necesarias para que se ajustaran a las necesidades propuestas, es decir: pelotas, señales de stop y diferentes escenarios en donde los dos objetos anteriores no existieran. De esta manera se busca etiquetar los tres objetos de interés, para que el robot móvil se detenga, se active o no realice ninguna acción.

Este conjunto de datos se dividió en tres categorías o clases principales que la red neuronal debía aprender a distinguir:

a. Figura de activación: Estas imágenes representan una figura que sea indicativa para que un vehículo móvil dé inicio a su actividad, se escogieron pelotas de forma aleatoria, por su simplicidad.
b. Letreros de Stop: Fotografías de señales de alto (stop) en diferentes posiciones.
c. Nada: Imágenes que representaban fondos vacíos como, por ejemplo, el suelo, paredes u otros escenarios donde no estuvieran presentes ni pelotas ni señales de alto. Esta clase es crucial para que la red pudiera aprender a identificar la ausencia de los objetos de interés.

Para cada una de estas tres clases, las imágenes se organizaron en tres subcarpetas destinadas a cumplir diferentes funciones durante el desarrollo y evaluación del modelo de la red neuronal: Carpeta de Entrenamiento (*train*), Carpeta de Validación (*valid*) y Carpeta de Pruebas (*test*). Estas carpetas son guardadas dentro de una computadora, la cual contiene el programa de MATLAB que será el encargado de procesar el entrenamiento y la ejecución de la red neuronal.

### *3.2. Creación de los contenedores y asignación de etiquetas*

Para cada conjunto de datos de entrenamiento, validación y prueba, se agruparon las imágenes en contenedores o *datastores* denominados *imageDatastore*. Un *imageDatastore* es un objeto que facilita el manejo de grandes volúmenes de imágenes. A estos *datastores* se les asignan etiquetas categóricas según la clase correspondiente.

### *3.3. Preprocesamiento de imágenes*

Para el tratamiento de las imágenes utilizadas, se realizó un procedimiento estandarizado. Este procedimiento estandarizado consiste en definir un tamaño específico para todas las imágenes que componen la base de datos, incluyendo los conjuntos de prueba y de entrenamiento; en nuestro trabajo se utilizó un tamaño de $100 \times 100$ píxeles. Se establecen los filtros para trabajar en escala de grises y RGB, además, se establece una normalización en los valores de las matrices de las imágenes para que tengan un valor entre 0 y 1.

Una vez que se realizó el almacenamiento y ajuste de las imágenes para realizar el entrenamiento de la red neuronal, se procesó dentro de MATLAB a través de la *librería Deep Learning Toolbox*.

Para mejorar la capacidad de generalización del modelo, se realizó un aumento en la base de datos obtenida. Para este aumento en la base de datos se aplican transformaciones aleatorias a las imágenes de entrenamiento, como rotación, desplazamiento, escalado y reflexión horizontal. Esto último se realiza mediante una función de aumento de la base de datos llamada *imageDataAugmenter*, después se utiliza la función *augmentedImageDatastore* para aplicar estas transformaciones dinámicamente al entrenar.

### *3.4. Arquitectura de la red neuronal*

Primero que nada, es necesario definir el número de clases de salida que la red tiene y, en función de ello, se define una capa de salida de 3 neuronas o clases; en la misma forma, para el número de entradas, es necesario definir el tamaño del vector de ingreso a la red neuronal, en este caso, sabemos que se van a ingresar imágenes de $100 \times 100 \times 3$ píxeles, donde $100 \times 100$ representa el tamaño de la imagen después del *resize* y el valor 3 representa las tres capas de color (RGB) que MATLAB da a las imágenes; para la extracción de características, esta arquitectura cuenta con 3 bloques convolucionales; sigue el aplanado; una capa completamente conectada; una capa *softmax*, la cual permite calcular la probabilidad de que los datos calculados pertenezcan a cada una de las clases de salida, esta es una función utilizada en redes multiclase; y finalmente la capa de salida. La arquitectura de esta red puede verse en la Figura 3, donde se presenta esta arquitectura en forma de bloques para facilitar su visualización.

Para especificar las opciones de entrenamiento primero se establecen los parámetros que controlan como se entrenará la

red neuronal. Como primer paso se define el algoritmo de optimización, la tasa de aprendizaje y su ajuste progresivo, el número de épocas, el tamaño de los lotes de datos y si los datos se reorganizan en cada época. Segundo, se indican los datos de validación, con qué frecuencia se evalúan y se habilita la visualización del progreso. Finalmente, se incluye una condición para detener el entrenamiento si la función de pérdida se vuelve inválida.

Como optimizador se utiliza *Adam* por ser uno de los más utilizados, además de proporcionar un entrenamiento rápido. Como es sabido se utiliza como primer momento el promedio del gradiente y como segundo el promedio del gradiente al cuadrado (RMSProp). Como fue mencionado anteriormente, se utilizó MATLAB en una máquina local; se remarca que la base de datos utilizada fue hecha con imágenes propias, tomadas específicamente para este proyecto. Se pueden ver los parámetros de entrenamiento en la Figura 4.

**Configuración del Entrenamiento**

**Optimizador Adam**

| | |
|---|---|
| **Algoritmo:** | Adam |
| **Beta1 (momento 1):** | 0.9 |
| **Beta2 (momento 2):** | 0.999 |
| **Epsilon:** | 1e-8 |

**Programación de Learning Rate**

| | |
|---|---|
| **Inicial:** | 0.001 |
| **Épocas 1-10:** | 0.001 |
| **Épocas 11-20:** | 0.0005 (×0.5) |
| **Épocas 21-30:** | 0.00025 (×0.5) |

**Validación y Monitoreo**

| | |
|---|---|
| **Frecuencia:** | Cada 30 mini-batches |
| **Métricas:** | Loss, Accuracy |
| **Criterio de Parada:** | Detección de NaN/Inf |

Figura 4. Parámetros definidos para la función de entrenamiento.

Una vez que se ha definido tanto la arquitectura de la red neuronal como sus parámetros de funcionamiento, es necesario iniciar el entrenamiento de esta. El conjunto de datos de la base de datos se divide en los conjuntos de entrenamiento, validación y prueba, en proporciones porcentuales de 63/24/13, respectivamente. Posteriormente, se utiliza el conjunto de prueba para verificar la precisión general del modelo realizado. Estos elementos se verán con mayor detalle más adelante.

### *3.5. Implementación del dispositivo móvil*

Una vez que se diseñó la arquitectura de la red neuronal y esta fue entrenada con la base de datos, el siguiente paso es probar el modelo en un entorno real, por lo cual se realiza la implementación física de un pequeño vehículo que será controlado por la CNN. Esto involucró el ensamblaje del dispositivo, la programación y montaje de la tarjeta de desarrollo ESP32-CAM que se encarga del control, la comunicación con la CNN, y la configuración del programa en MATLAB que actúa como servidor y procesador de imágenes.

Para el robot móvil, se utilizó una estructura sobre la cual se montaron los componentes electrónicos. Dentro de esta estructura se colocaron diferentes módulos tales como el módulo de control de dirección, el cual es dirigido por medio de un puente H (específicamente el L298N), el cual controla los movimientos de las ruedas del dispositivo; y el módulo de comunicación, gestionado por medio de la tarjeta ESP32-CAM, la cual se conecta al puente L298N para realizar el intercambio de comandos de control.

Específicamente, se utilizaron cuatro pines de la ESP32-CAM para determinar el sentido de giro de cada motor (dos por motor) y otros dos pines con capacidad PWM (Modulación por Ancho de Pulso) para regular la velocidad de los motores, permitiendo así movimientos controlados como avanzar, girar o detenerse. La ESP32-CAM es la encargada de capturar las imágenes del entorno y gestionar la comunicación inalámbrica por medio de Wi-Fi. En la Figura *5* se puede ver el diseño del circuito implementado.

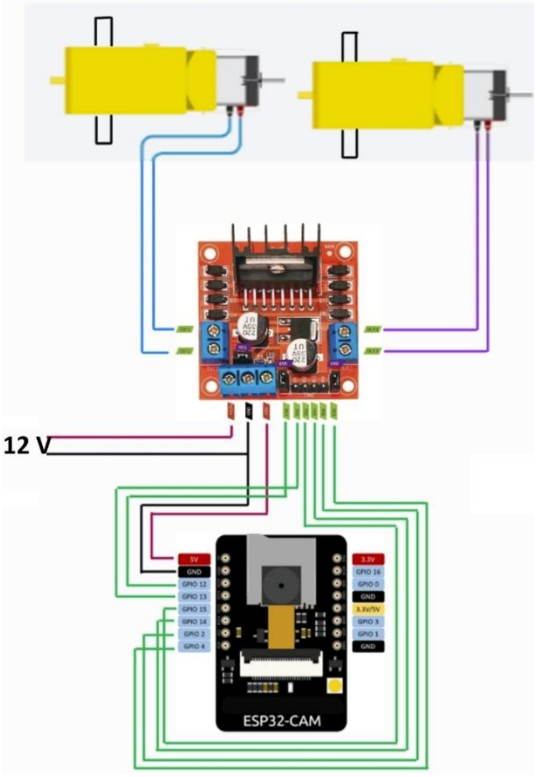


Figura 5. Esquema de la arquitectura utilizada.

### *3.6. Programación ESP32-CAM*

El programa de control del ESP32-CAM fue desarrollado en *MicroPython*, un lenguaje que está diseñado específicamente para potenciar microcontroladores y microprocesadores. El script dentro del ESP32-CAM se encarga de conectarla a la red Wi-Fi e inicializa el módulo de la cámara, ajustando parámetros como la resolución, el formato y la calidad, buscando que la velocidad de transmisión no se vea menguada.

El punto central de la comunicación se define en función de la configuración de un servidor socket TCP/IP, haciendo notar que, para realizar las conexiones inalámbricas, es necesario definir una dirección IP y el puerto de conexión para la ESP32-CAM.

Este servidor se queda en un estado de alerta dentro de un puerto específico (previamente definido), esperando la señal de una conexión de MATLAB. Cuando MATLAB se conecta, la ESP32-CAM captura una imagen y la envía de vuelta a MATLAB. Después, cierra la conexión y espera una nueva conexión para recibir el resultado de la clasificación (“pelota”, “stop”, “nada”) desde MATLAB.

El script en MATLAB actúa como el cliente del socket de la ESP32-CAM para recibir imágenes y, posteriormente, para enviar los resultados de la clasificación, además de realizar el procesamiento de las imágenes con la CNN entrenada.

Dentro de este sistema son considerados umbrales de confianza para cada una de las clases de salida, esto se realiza con la intención de determinar una fiabilidad mínima sobre las respuestas en la salida. Dentro de todo el sistema, también se considera una interfaz visual en donde se puede percibir la imagen que ha sido captada, la cantidad de veces que ha sido detectada (conteo acumulativo) y la confianza de la detección en tiempo real.

Una forma de resumir el comportamiento general del sistema es: La ESP32-CAM es conectada y recibe la imagen del entorno; el sistema verifica la imagen y sus características; esta es enviada a la red neuronal generada y se realiza el pre procesamiento de la información, es decir, cambiar tamaño, normalización, etc.; esta imagen, ya preprocesada, es enviada a la CNN entrenada para ser clasificada y verificada bajo los umbrales de confianza definidos; el resultado de la clasificación (Pelota, stop o nada) es enviado nuevamente a la ESP32-CAM a través de una nueva conexión socket; a la par de esa transmisión, el gráfico de la interfaz de MATLAB se actualiza con los datos de clasificación y de ingreso; con la información recibida de la CNN, el vehículo realiza la acción de control encendiéndose apagándose o manteniéndose estático.

## 4. Análisis de resultados

Al comenzar el entrenamiento, se observa la cantidad de imágenes. La Figura 6 muestra el número de imágenes con los que se entrenó y validó la CNN.

```
### RESUMEN DE DATOS ###
Encontradas 473 imágenes de entrenamiento (242 pelotas, 131 stop, 100 nada).
Encontradas 180 imágenes de validación (136 pelotas, 19 stop, 25 nada).
Encontradas 101 imágenes de prueba (68 pelotas, 8 stop, 25 nada).
Iniciando entrenamiento de la red...
```

Figura 6. Cantidad de imágenes utilizadas para el entrenamiento y prueba de la red neuronal generada.

La gráfica del progreso del entrenamiento (Figura 7) da pistas de cómo aprendió la red. La precisión de entrenamiento subió rápido al inicio y se estabilizó después de las primeras 200 iteraciones. Esto dice que la red entendió rápido las características principales. Por otro lado, la precisión de validación, que dice si la red funciona bien con imágenes nuevas, se mantuvo entre el 80 y el 90 por ciento durante el correr de las épocas. Al final del entrenamiento, después de 80 épocas y 14 iteraciones por época, la red neuronal tuvo una precisión final de entrenamiento del 86.67 por ciento.

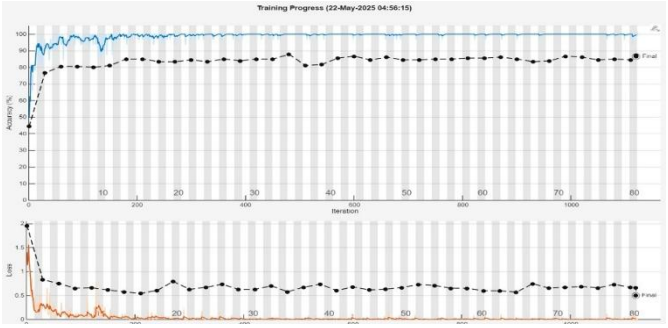


Figura 7. Proceso gráfico del entrenamiento de la red neuronal.

Cuando la red ya estaba entrenada y guardada, se probó con las imágenes de *prueba*. Aquí, el sistema logró una precisión general del 88.12 por ciento al clasificar todas estas imágenes de prueba.

Al ver los ejemplos de las predicciones que hizo el modelo (Figura 8), es posible ver que a la red le costaba identificar bien las pelotas cuando estaban lejos o cuando el fondo de la imagen era predominante. En los casos mostrados, una pelota lejana se podía parecer a las fotos de "nada" que se usaron para entrenar.

Pero si los objetos estaban cerca y se veían claros, la detección funcionaba bien.

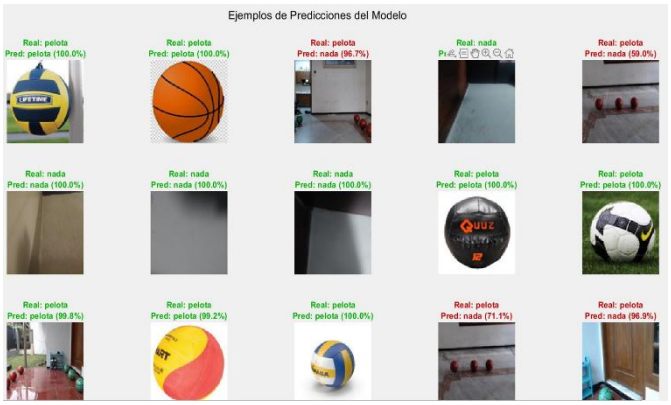


Figura 8. Verificación del reconocimiento de la clase “pelota” dentro de la red neuronal.

En la matriz de confusión, mostrada en la Figura 9, se puede visualizar que la mayoría de los errores que se presentan cuando se clasifican imágenes con pelotas, se malinterpreta la clase “pelota” con la clase “nada”. Esto ya se había definido ya que, cuando las pelotas se encuentran a una distancia alejada, el CNN está identificando como si fuera un entorno “general”, por lo que tiene problemas de identificación en este aspecto. Para la interpretación visual del funcionamiento del sistema propuesto, se generó una interfaz gráfica, la cual muestra información relacionada a la captura de objetos y respuesta de la CNN en tiempo real. La interfaz está compuesta de tres secciones principales:

A. Visualización de la Cámara: En la parte superior de la ventana, se muestra la imagen que la ESP32-CAM está capturando y transmitiendo en ese momento. El título de esta imagen se actualiza dinámicamente para indicar la clase detectada por la red neuronal (“Pelota”, “Stop” o “Nada”), el porcentaje de confianza de esa detección y una

breve descripción de la acción que el robot debería estar realizando.

B. Gráfica de Detecciones Acumuladas: Ubicada en la parte inferior izquierda, esta gráfica de barras presenta un historial del número total de veces que cada objeto ha sido detectado desde que se inició el programa. La primera barra corresponde a las “Pelotas”, la segunda a los letreros de “Stop” y la tercera a “Nada”.

C. Gráfica de Confianza de la Última Detección: En la parte inferior derecha, otra gráfica de barras muestra los porcentajes de confianza que la red neuronal asignó a cada una de las tres clases para la imagen más reciente procesada. La altura de cada barra para “Nada”, “Pelota” y “Stop” (en ese orden) indica qué tan segura está la red de que esa clase específica está presente en la imagen actual.

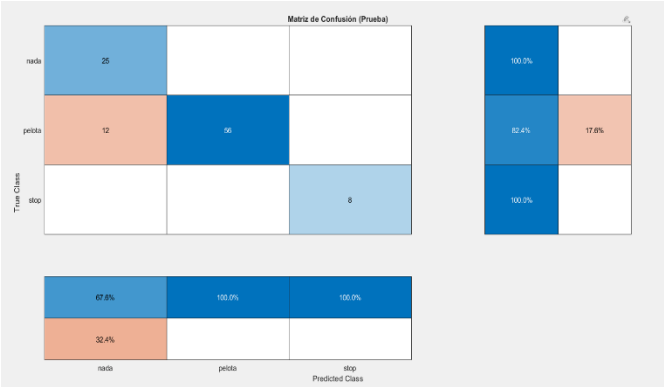


Figura 9. Matriz de confusión. Compara los resultados obtenidos con los esperados y divide en porcentajes los resultados correctos e incorrectos.

Finalmente, en las pruebas con el control del dispositivo, se observó lo que las gráficas ya nos decían. El robot hacía bien sus movimientos cuando las pelotas o los letreros de stop estaban cerca de su cámara. Si los objetos estaban muy lejos o medio escondidos, la detección no era tan buena y el dispositivo a veces no hacía lo esperado. En la figura 10 y en la Figura 11 se muestra cómo se presentaban las imágenes al carro.

En las Figuras 12, 13 y 14 se pueden observar ejemplos concretos de la interfaz durante la detección de cada uno de los objetos de interés y cuando no se detecta ninguno.

En función de las muestras obtenidas, podemos decir que el comportamiento de nuestro sistema de visión artificial es coherente con respecto a las condiciones de entrenamiento. Como ya se ha mencionado con anterioridad, debido a que las imágenes fueron tomadas personalmente y, en comparación con sistemas de este tipo, se tuvo una cantidad limitada de las mismas, el conjunto de datos creado es funcional; aunque puede verse afectada su eficacia en la respuesta de salida debido principalmente a que se tiene una limitada variedad de imágenes en cuanto a escenarios, condiciones de iluminación y distancias. Esto puede ser mejorado, y dejado como trabajo futuro, considerando las siguientes acciones:

a. Ampliar y diversificar el conjunto de imágenes que han sido utilizadas para entrenar y validar la CNN. Obviamente, generando diversificación en la forma de tomar las fotografías, es decir, ampliar el conjunto de entornos, los ángulos de captura, las condiciones lumínicas, uso de sombras, existencia de imágenes más saturadas en objetos, fondos complejos. De esta forma se lograría una generalización más adecuada.

b. Otra posible opción es la realización de procesamiento de las imágenes, obteniendo filtrados y contrastes de las mismas.

c. Uso de redes previamente entrenadas con grandes volúmenes de datos (*transfer learning*) y luego realizar la adaptación a la *dataset* generada en este proyecto. Sin embargo, esa no era la intención de este proyecto, la intención fue crear la base de datos desde el inicio y verificar la respuesta de la misma ante el aprendizaje por medio de una CNN.


Figura 10. Pruebas del vehículo en funcionamiento. Se muestra un letrero de "stop" al carro y la red neuronal lo detecta exitosamente.

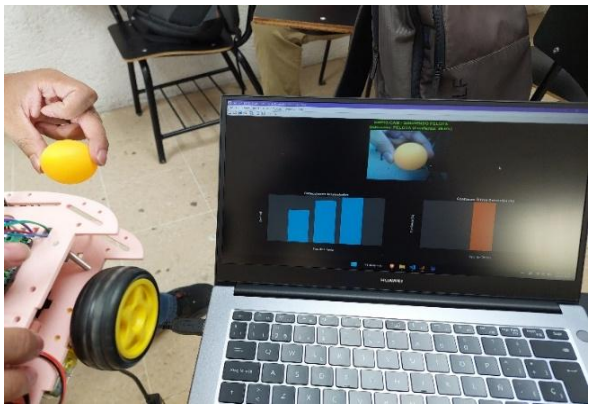

Figura 11. Pruebas del vehículo en funcionamiento. Se muestra una pelota al carro y la red neuronal lo detecta exitosamente.

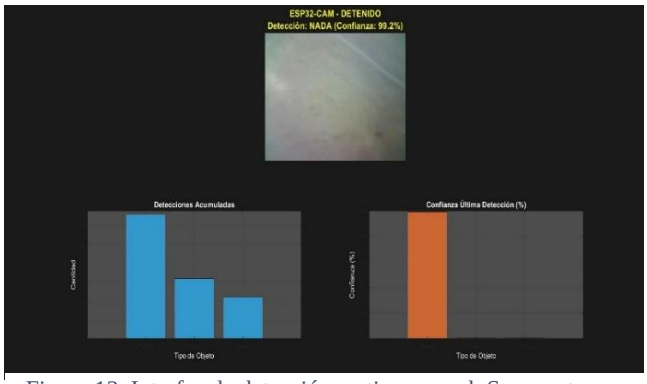


Figura 12. Interfaz de detección en tiempo real. Se muestra un entorno sin los objetos de interés. El sistema clasifica correctamente la escena como "Nada", lo cual se refleja en las gráficas de confianza.

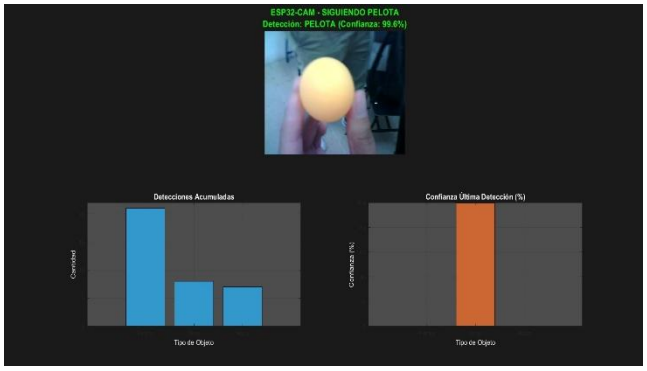


Figura 13. Interfaz de detección en tiempo real. Se muestra la pelota capturada. El sistema clasifica correctamente la escena como "Pelota", lo cual se refleja en las gráficas de confianza.


Figura 14. Interfaz de detección en tiempo real. Se muestra el stop capturado. El sistema clasifica correctamente la escena como "Stop", lo cual se refleja en las gráficas de confianza.

## 5. Conclusiones

En este artículo se presentó el desarrollo de un sistema de activación y desactivación de un robot móvil utilizando visión artificial. Para el correcto funcionamiento del proyecto, se construyó la base de datos que alimentó a una red neuronal convolucional en MATLAB para que esta pudiera identificar los objetos que hacen funcionar el vehículo.

Para el funcionamiento del sistema se utilizaron tres clases, una clase para activar el vehículo (Pelota), una para detenerlo (Stop) y una como punto neutral (Nada), es decir, no se realiza ninguna acción.

La red neuronal en funcionamiento alcanzó una exactitud de entre 85% y 90% en las diferentes pruebas realizadas, lo cual se puede considerar como un buen resultado ya que se utilizó para el entrenamiento una base de datos creada desde cero, la cual, para los estándares modernos, puede considerarse pequeña. Dicho de otra forma, a pesar de la cantidad de datos, el resultado obtenido alcanza un alto porcentaje de reconocimiento.

El sistema de visión artificial permitió generar una respuesta autónoma del vehículo. Es importante mencionar que este sistema fue manejado de forma remota, es decir, el vehículo capta el entorno, envía la información vía inalámbrica a una red neuronal hecha con MATLAB, la red procesa los datos, los muestra en el sistema gráfico de identificación y envía la señal de regreso al vehículo para que este realice la tarea asignada. Todo este proceso se realizó de forma adecuada.

Se ha de mencionar que, debido a que la base de datos de entrenamiento se creó desde cero, se tuvieron problemas con el correcto reconocimiento, sin embargo, la eficacia de respuesta de la red no fue baja; por el contrario, muestra una eficacia de respuesta elevada.

Un posible trabajo futuro consiste en generar una base de datos más completa y que pueda ser mejor interpretada por la red neuronal.

Todo el procesamiento de la red neuronal se llevó a cabo fuera del dispositivo móvil, permitiendo que este se enfocara únicamente en capturar los elementos visuales y responder a los mismos, evitando un costo de procesamiento interno. El procesamiento se ejecutó dentro de una computadora externa al dispositivo, por lo que, como trabajo futuro, este procesamiento se realizará en la nube de MATLAB para verificar la viabilidad y los tiempos de respuesta.

**Agradecimientos**

Se agradece a los proyectos PAPIME PE107225 de la UNAM y PICI 2410 de la FESC.

## Referencias

Almusawi, H. A., Al-Jabali, M., Khaled, A. M., Péter, K., & Géza, H. (2022, October). Self-Driving robotic car utilizing image processing and machine learning. In IOP Conference Series: Materials Science and Engineering (Vol. 1256, No. 1, p. 012024). IOP Publishing.

Amer, K., Samy, M., Shaker, M., & ElHelw, M. (2021, January). Deep convolutional neural network based autonomous drone navigation. In Thirteenth International Conference on Machine Vision (Vol. 11605, pp. 16-24). SPIE.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... & Zieba, K. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

Cahyono, F. Y. A., Suharto, N., & Mustafa, L. D. (2022). Design and build a home security system based on an esp32 cam microcontroller with telegram notification. Journal of Telecommunication Network (Jurnal Jaringan Telekomunikasi), 12(2), 58-64.

Coşkun, M., Uçar, A., Yildirim, Ö., & Demir, Y. (2017, November). Face recognition based on convolutional neural network. In 2017 international conference on modern electrical and energy systems (MEES) (pp. 376-379). IEEE.

Elhattab, K., Abouelmehdi, K., & Elatar, S. (2023, October). New model to monitor plant growth remotely using esp32-cam and mobile application. In 2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 1-6). IEEE.

Jain, A. K. (2018, March). Working model of self-driving car using convolutional neural network, Raspberry Pi and Arduino. In 2018, Second International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1630-1635). IEEE.

Lu, J., Tang, S., Wang, J., Zhu, H., & Wang, Y. (2019, June). A review on object detection based on deep convolutional neural networks for autonomous driving. In 2019 Chinese Control and Decision Conference (CCDC) (pp. 5301-5308). IEEE.

Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., & Valenzuela, A. (2020). A review of convolutional neural network applied to fruit image processing. Applied Sciences, 10(10), 3443.

Prathapagiri, D., & Kosalendra, E. (2021). Wi-Fi door lock system using ESP32 CAM based on IoT. The International journal of analytical and experimental modal analysis, 13(7), 2000-2003.

Raju, N., Navya, A., Koteswaramma, N., Mounika, B., & Rajeshwari, T. (2022). IoT-based Door Access Control System using ESP32 CAM. International Journal of Engineering Inventions, 11(12), 9-15.

Rausch, V., Hansen, A., Solowjow, E., Liu, C., Kreuzer, E., & Hedrick, J. K. (2017, May). Learning a deep neural net policy for end-to-end control of autonomous vehicles. In 2017 American control conference (ACC) (pp. 4914-4919). IEEE.

Rusimamto, P. W., Endryansyah, L. A., Harimurti, R., & Anistyasari, Y. (2021). Implementation of arduino pro mini and ESP32 cam for temperature monitoring on automatic thermogun IoT-based. Indones. J. Electr. Eng. Comput. Sci, 23(3), 1366-1375.

Sadeghi Esfahlani, S., Sanaei, A., Ghorabian, M., & Shirvani, H. (2022). The deep convolutional neural network role in the autonomous navigation of mobile robots (SROBO). Remote Sensing, 14(14), 3324.

Sainath, V., & Reddy, S. (2021, August). Deep learning for autonomous driving system. In 2021, Second International conference on electronics and sustainable communication systems (ICESC) (pp. 1744-1749). IEEE.

Sakhare, K. V., Tewari, T., & Vyas, V. (2020). Review of vehicle detection systems in advanced driver assistant systems. Archives of Computational Methods in Engineering, 27(2), 591-610.

Sanango Tufiño, A. A. (2022). Desarrollo de un sistema de visión artificial basado en redes convolucionales para la detección de las señales de tránsito implementado sobre un vehículo autónomo.

Seth, A., James, A., & Mukhopadhyay, S. C. (2020). 1/10th scale autonomous vehicle based on convolutional neural network. International Journal on Smart Sensing and Intelligent Systems, 13(1), 1-1Shoeb, M., Ali, M. A., Shadeel, M., & Bari, D. M. A. (2022). Self-driving car: using Opencv2 and machine learning. The International journal of analytical and experimental modal analysis (IJAEMA), ISSN, 0886-9367.

Williams, H. A., Jones, M. H., Nejati, M., Seabright, M. J., Bell, J., Penhall, N. D., ... & MacDonald, B. A. (2019). Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms. biosystems engineering, 181, 140-156.