Asociación Colombiana de Facultades de Ingeniería

Revista Digital Educación en Ingeniería
ISSN 1900 - 8260

AI_Card en el borde: para la enseñanza-aprendizaje de redes neuronales e inteligencia artificial

José Ferney Cortés Garnica [a], Kevin Daniel Martínez Ospino [b]

[a] *Secretaría de Educación Distrital, Bogotá, Colombia*
[b] *Politécnico Santafé de Bogotá, Colombia*
ferneycortes@gmail.com, kevindanielmartinezospino@gmail.com

Resumen— Este artículo reporta una prueba de concepto desarrollada con dos muestras intencionales —seis estudiantes de grado undécimo sin formación previa en programación y dieciocho profesionales de distintas áreas— y declara tres contribuciones: (1) el diseño y la construcción de un hardware educativo de bajo costo que integra en un único módulo el microcontrolador ESP32-S3, cámara DVP, punto de acceso Wi-Fi, microSD y trece salidas GPIO para actuadores; (2) el desarrollo de D_T_I_AI.py, un software unificado que ejecuta de forma local y sin conexión a Internet el pipeline completo Datos–Entrenamiento–Inferencia, entrena y compara automáticamente tres arquitecturas (MLP, CNN y CNN con regularización avanzada) e incorpora un módulo de análisis multimodal con modelos de lenguaje multimodal (LMM) servidos por Ollama; y (3) los hallazgos preliminares de la prueba de concepto, que sugieren que participantes sin formación previa en programación logran completar el ciclo de construcción de data-sets, entrenamiento, contraste por inferencia y control de actuadores físicos dentro de sesiones acotadas de trabajo.

Palabras Clave— Redes neuronales, LMM, inteligencia artificial en el borde, data set, entrenamiento, inferencia.

Recibido: 2 de julio de 2026. Revisado: 29 de julio de 2026. Aceptado: 31 de julio de 2026.

AI_Card on the edge: for teaching and learning neural networks and AI

Abstract— This article reports a proof of concept carried out with two purposive samples —six eleventh-grade students with no prior programming background and eighteen professionals from various fields— and states three contributions: (1) the design and construction of low-cost educational hardware integrating an ESP32-S3 microcontroller, DVP camera, Wi-Fi access point, microSD and thirteen GPIO outputs in a single module; (2) D_T_I_AI.py, unified software that runs the complete Data–Training–Inference pipeline locally and without an Internet connection, automatically trains and compares three architectures (MLP, CNN and CNN with advanced regularisation), and embeds a multimodal analysis module using multimodal-language models (LMM) served by Ollama; and (3) preliminary findings suggesting that participants without prior programming background can complete the full cycle of dataset construction, training, inference-based contrast and physical actuator control within bounded working sessions.

Keywords— Neural networks, LMM, AI on the edge, dataset, training, inference,

1 Introducción

1.1 Inteligencia artificial y educación

La relación inteligencia artificial (IA) educación se ha venido abordando desde 2019 con el consenso de Beijín: "Sobre la inteligencia artificial y la educación" [1] y desde la UNESCO en su documento "IA y educación: guía para las personas a cargo de formular políticas" [2]. En 2025 se resalta la política educativa global con la publicación, en el marco de la Semana del aprendizaje digital de la UNESCO, de dos marcos de competencias complementarios: el Marco de competencias en inteligencia artificial para docentes [3] —que plantea la necesidad de que los maestros vayan más allá del dominio técnico de las herramientas y desarrollen una comprensión profunda de los principios éticos, los fundamentos matemáticos y las implicaciones sociales de los sistemas de inteligencia artificial—; por otro lado, está el Marco de competencias en inteligencia artificial para estudiantes [4], el cual organiza 12 competencias en cuatro dimensiones: mentalidad centrada en el ser humano (que incluye pensamiento crítico, ética y ciudadanía digital), ética de la inteligencia artificial (privacidad, sesgo, transparencia y responsabilidad), técnicas y aplicaciones de inteligencia artificial (fundamentos de aprendizaje automático, visión por computadora, procesamiento de lenguaje natural), y diseño de sistemas de IA (metodología de proyectos, evaluación y mejora de sistemas). Cada competencia se trabaja en tres niveles de progresión: comprender, aplicar y crear.

A nivel Nacional, en febrero de 2025 aparece [5], trabajo que presenta la Política Nacional de inteligencia artificial cuyo objetivo es generar capacidades para la investigación y el desarrollo, la adopción y el aprovechamiento ético sostenible de inteligencia artificial con el fin de impulsar la transformación social y económica de Colombia.

El documento presenta un proyecto orientado a incorporar habilidades y competencias digitales desde la educación básica hasta la superior para facilitar la comprensión y participación de todos en el uso y aprovechamiento de la inteligencia artificial, esto a raíz de lo encontrado por la misión de sabios 2020, donde recomiendan reformar la educación en áreas de ciencias, tecnología, ingeniería y matemáticas (STEM) para desarrollar competencias en inteligencia artificial y otras tecnologías digitales que son cada vez más necesarias en el sector productivo. Por lo anterior, el Departamento Nacional de Planeación, el Ministerio de Tecnologías de la Información y las Comunicaciones y el Ministerio de Ciencia, Tecnología e Innovación recomiendan al Consejo Nacional de Política

© Los autores; licenciado a ACOFI.

Económica y Social diseñar y ejecutar, junto con el Ministerio de Educación Nacional, una Misión para la Transformación Digital de la Educación que genere acciones y recomendaciones encaminadas a la incorporación de las tecnologías digitales, especialmente la inteligencia artificial, en el sistema educativo.

1.2 Inteligencia artificial en el borde

La inteligencia artificial en el borde o *AI on the Edge* es la inteligencia artificial que se ejecuta localmente en sistemas embebidos en el borde; los sistemas embebidos, según [6], son los computadores que controlan la electrónica de sistemas físicos como sensores, cámaras, máquinas industriales y que pueden ser dispositivos simples como el microcontrolador que controla una pantalla digital o sofisticados como el computador Linux que controla el Smart TV. El borde se refiere a que estos dispositivos se pueden conectar en red unos con otros; al final se tiene una IA que corre en estos dispositivos en lugar de la nube. Este enfoque procesa datos donde se generan, permitiendo decisiones instantáneas sin demoras de red.

La inteligencia artificial en el borde agrupa dispositivos que ejecutan inferencia y a veces entrenamiento de modelos de IA localmente, sin depender de la nube. Se clasifican habitualmente en tres niveles según su potencia computacional:

- Nivel 3: aceleradores Edge de alto rendimiento como la NVIDIA Jetson Orin Nano, Google Coral y Raspberry Pi + Hailo que cuentan con GPU o una unidad de procesamiento neuronal o NPU. El representante más importante de esta categoría en la familia NVIDIA Jetson Orin cuyos módulos entregan entre 20 TOPS (Orin Nano 4 GB) y 275 TOPS (AGX Orin 64 GB) es el Jetson Orin Nano Super Developer Kit que alcanza 67 TOPS de rendimiento INT8 gracias a su GPU NVIDIA Ampere con 1024 núcleos CUDA y 32 Tensor Cores, combinada con 8 GB de memoria LPDDR5 de alta velocidad y 102 GB/s de ancho de banda colocando a la Jetson Orin Nano en la misma categoría, aunque en la posición de entrada, que los aceleradores de inferencia de centros de datos.
- Nivel 2: agrupa a las computadoras de placa única, corren un sistema operativo completo, generalmente Linux embebido o Android sobre procesadores de arquitectura ARM de 64 bits con varios núcleos a frecuencias de entre 1 GHz y 2 GHz, y cuentan con memorias RAM de 1 GB a 8 GB. Raspberry Pi, en especial el modelo 4 (ARM Cortex-A72, 4–8 GB de RAM, 5–7 W) y el modelo 5 (ARM Cortex-A76, hasta 8 GB, consumo entre 5 y 12 W). Estos dispositivos son capaces de ejecutar inferencia con modelos de tamaño medio —CNN como MobileNet, EfficientNet-Lite y YOLOv5 Nano— a tasas de entre 5 y 30 fotogramas por segundo en CPU, lo que los hace adecuados para tareas de visión por computadora en tiempo casi real como la detección de objetos en cámaras de seguridad educativas, sistemas de reconocimiento facial en control de acceso, o asistentes de laboratorio con visión artificial.
- Nivel 1: constituido por los microcontroladores de ultra bajo consumo sobre los cuales se ejecutan modelos de aprendizaje automático compactos, un campo conocido como Tiny Machine Learning (TinyML). Estos dispositivos entre los que se encuentran el ESP32-S3, el Arduino Nano 33 BLE Sense, el STM32 de STMicroelectronics y el Microchip ATSAMD51 corren directamente sobre el metal o sobre sistemas operativos de tiempo real (RTOS) de huella mínima sin la presencia de un sistema operativo de propósito general como Linux. Sus núcleos de procesamiento operan entre 48 MHz y 240 MHz, cuentan con memorias RAM de entre 256 KB y 8 MB, y su consumo energético típico durante la inferencia se sitúa entre 5 mW y 500 mW. Esta eficiencia energética extrema permite alimentarlos con baterías de botón durante meses o incluso años, lo que los convierte en la opción natural para sistemas portátiles, wearables, sensores remotos y dispositivos de Internet de las Cosas (IoT) de bajo costo.

La AI_Card desarrollada en este proyecto pertenece a este nivel, se construyó sobre el microcontrolador ESP32-S3 (Xtensa LX7 a 240 MHz, con instrucciones vectoriales para operaciones de redes neuronales), e integra cámara DVP, Wi-Fi en modo Access Point, microSD y 13 pines GPIO de salida en un único módulo. Su pipeline completo —recolección de imágenes, entrenamiento de tres arquitecturas de redes neuronales (MLP, CNN simple y CNN con regularización avanzada) y clasificación en tiempo real con control de actuadores físicos se ejecuta sin ninguna dependencia de Internet.

1.3 Inteligencia artificial en el borde y la educación

Para contextualizar al lector, es preciso anotar que en 2020 los autores de [7] trabajaron el aprendizaje profundo en computadoras pequeñas (TinyML). Allí hicieron recolección de datos, entrenamiento, conversión del modelo e inferencia, abordado desde el paradigma del aprendizaje. Para 2025, los autores de [8] plantean cómo la inteligencia artificial en el borde y la computación de borde (o *Edge Computing* en inglés) está transformando la educación moderna al habilitar procesamiento de datos en tiempo real y local; adicionalmente, en estos trabajos se muestran las ventajas relacionadas con la retroalimentación en tiempo real, el acceso, la independencia de la nube y la conexión. Allí se exponen problemáticas para las instituciones educativas relacionadas con la privacidad y seguridad de los datos, costos de plataformas, APIs y otros servicios, la rentabilidad, dependencia de proveedores de servicios, la complejidad de gestionar múltiples servicios y plataformas, limitaciones de la nube, control limitado de recursos en función de los costos para las instituciones educativas, curvas de aprendizaje para nuevas plataformas, entre otros. Tras un análisis del panorama de la inteligencia artificial en el borde en instituciones educativas, se plantearon beneficios relacionados con las experiencias de aprendizaje personalizadas y adaptadas en tiempo real, la compatibilidad que ofrecen los dispositivos *AI on the Edge* con los existentes y la no dependencia de la actualización de equipos y redes en las instituciones. También, afirman que existirán cada vez más modelos eficientes de inteligencia artificial y muchos serán adaptados para dispositivos en el borde. Al final concluyen que los costos de implementación de *AI Edge* en las instituciones dependerá de sus presupuestos. Cabe anotar que las plataformas de programación por bloques y los servicios de entrenamiento en la nube han cumplido un papel decisivo en la democratización del primer contacto escolar con la inteligencia artificial, y su valor didáctico está ampliamente documentado: reducen la barrera de entrada, permiten obtener un clasificador

funcional en minutos y hacen visible el resultado del aprendizaje automático sin requerir programación.

Por su diseño, sin embargo, estas metodologías priorizan el rol del estudiante como usuario de modelos pre entrenados sobre el rol de creador tecnológico: el data-set, la arquitectura de la red, el bucle de entrenamiento y el despliegue permanecen encapsulados tras una interfaz, y el ciclo suele cerrarse en la pantalla, sin acoplamiento con el mundo físico. Con todo, la AI_Card aquí presentada no se propone como sustituto de esas herramientas, ni como objeción a su uso, sino como una alternativa complementaria que desplaza el foco hacia el rol de creador: el participante decide qué imágenes componen su data-set, observa el efecto de esa decisión en los gráficos de entrenamiento, compara arquitecturas, contrasta el modelo por inferencia y ve cómo su clasificador acciona un LED, un servo o un motorreductor. En esa medida, la propuesta resulta disruptiva menos por su tecnología que por el lugar que asigna al estudiante dentro del ciclo de vida del modelo.

1.4 Plataformas comparables para la enseñanza de TinyML

Las opciones habituales para llevar TinyML al aula son de tres tipos. Las primeras combinan una placa comercial con una plataforma de entrenamiento en la nube, como el kit Arduino Tiny Machine Learning con Edge Impulse, esquema sobre el que se han estructurado cursos masivos y redes académicas internacionales. Las segundas emplean microcontroladores genéricos como el RP2040 del Raspberry Pi Pico con TensorFlow Lite para Microcontroladores, lo que exige un flujo de trabajo más artesanal y conocimientos de C++. Las terceras suben de nivel de cómputo hacia computadores de placa única o aceleradores, con el consiguiente aumento de costo y consumo. La Tabla 1 compara estas alternativas con la AI_Card en las dimensiones que resultan críticas para una institución educativa oficial: dependencia de conectividad y de cuentas externas, alcance del ciclo cubierto, disponibilidad inmediata de salidas físicas y costo por puesto de trabajo.

La diferencia con el desarrollo aquí descrito es que reivindica no el poder de cómputo (donde la AI_Card comparte nivel con las placas de referencia), sino la integración. El ciclo Datos–Entrenamiento–Inferencia se completa en un único artefacto y en un único software, sin registro en servicios externos, sin conectividad y con trece salidas “conectorizadas” listas para accionar actuadores, a lo que se añade una capa de análisis multimodal con LMM locales que no está presente en las plataformas educativas revisadas.

Tabla 1
Comparación de la AI_Card con plataformas para TinyML educativo

Plataforma	Nivel	Entrenamiento	Requiere internet / cuenta	Salidas físicas listas	Capa LMM local	Costo aprox. por puesto (COP)
AI_Card (ESP32-S3 + cámara DVP + indicador CMD + shield motores)	1	Local en el PC host (TensorFlow/Keras); tres arquitecturas comparadas automáticamente	No	13 GPIO conectorizados + shield de motores	Sí (Ollama)	205.100
Arduino Nano 33 BLE Sense + TinyML Kit (OV7675) + Edge Impulse	1	En la nube (Edge Impulse Studio)	Sí, ambas	GPIO sin conectorizar; requiere shield de motores y cableado adicional	No	230.000
Raspberry Pi Pico (RP2040) + cámara + TensorFlow Lite Micro	1	Local, flujo manual en C++	No, pero exige toolchain	GPIO sin conectorizar; requiere shield de motores y cableado adicional	No	100.000
Seeed XIAO ESP32S3 Sense + Edge Impulse	1	En la nube	Sí, ambas	GPIO sin conectorizar; requiere shield de motores y cableado adicional	No	150.000
Raspberry Pi 5 8GB (no incluye cámara)	2	Local, posible en el propio dispositivo	No	GPIO sin conectorizar; requiere shield de motores y cableado adicional	Modelos pequeños	1.024.000
NVIDIA Jetson Orin Nano	3	Local con GPU	No	GPIO sin conectorizar; requiere shield de motores y cableado adicional	Sí	2.505.150

Fuente: los autores.

1.5 Problema, preguntas y objetivos

Las alternativas disponibles para llevar el ciclo completo del aprendizaje automático al aula colombiana suponen, en distinta medida, conectividad estable, cuentas en servicios externos, transferencia de datos de los estudiantes a infraestructura de terceros o presupuestos por dispositivo difíciles de sostener en la educación media oficial. Falta evidencia sobre si un dispositivo de nivel 1, de bajo costo y enteramente desconectado, puede sostener ese ciclo completo —incluidos el entrenamiento y la actuación física— en condiciones reales de aula y con participantes sin formación previa en programación.

Frente a lo anterior es natural preguntarse: ¿es viable ejecutar el ciclo completo de recolección de datos, entrenamiento, inferencia y control de actuadores en una plataforma de nivel 1 de bajo costo, sin conexión a Internet ni GPU dedicada, dentro de tiempos compatibles con una sesión de clase?; ¿qué decisiones toman los participantes sin formación previa en programación cuando construyen sus propios data-sets, comparan arquitecturas de red y contrastan los modelos por inferencia con la AI_Card?; y ¿en qué medida una sesión única de una hora, mediada por pares previamente formados, permite a participantes noveles completar el pipeline Datos–Entrenamiento–Inferencia e interactuar con un modelo de lenguaje visual local?

El objetivo general es evaluar, mediante una prueba de concepto, la viabilidad técnica y la pertinencia pedagógica de la AI_Card como plataforma de IA en el borde para la enseñanza y el aprendizaje de redes neuronales. Los objetivos específicos son: (a) caracterizar el desempeño del pipeline en términos de exactitud de validación, tiempos de entrenamiento y latencia de respuesta de los modelos de lenguaje visual locales; (b) documentar los procedimientos y decisiones de los participantes durante la construcción de data-sets y el entrenamiento de modelos; y (c) identificar las condiciones operativas (duración, acompañamiento y materiales) bajo las cuales participantes noveles completan el pipeline.

Este trabajo corresponde a un estudio exploratorio de tipo prueba de concepto, con un diseño descriptivo de casos múltiples y enfoque predominantemente cualitativo apoyado en registros cuantitativos del sistema. No constituye un diseño experimental ni cuasiexperimental, no incorpora grupo de comparación ni medición pre-post del aprendizaje, y sus resultados no son estadísticamente generalizables.

2 Metodología

La metodología de este trabajo comprende dos componentes de naturaleza distinta. El primero es el desarrollo tecnológico de la plataforma —arquitectura de hardware, pipeline de software y decisiones de diseño—, descrito en 2.2 y 2.3. El segundo es una prueba de concepto de su uso educativo, ejecutada con dos muestras intencionales durante el primer semestre de 2026 y descrita en 2.1, 2.5 y 2.6. El componente educativo se declara formalmente como estudio piloto exploratorio: su propósito es establecer la viabilidad operativa de la plataforma y documentar el comportamiento de los participantes, no estimar efectos de aprendizaje ni comparar tratamientos. Las afirmaciones pedagógicas de las secciones 3 y 4 deben leerse bajo ese alcance.

Se presenta la implementación de la AI_Card para la enseñanza y el aprendizaje de redes neuronales e IA, construida para ejecutar pipelines completos de visión por computadora: recolección de datos, entrenamiento de redes neuronales convolucionales (CNN) y perceptrones multicapa (MLP), inferencia en tiempo real, y control de actuadores físicos mediante GPIO junto con un módulo de análisis multimodal basado en LMMs ejecutados localmente a través de Ollama, sin dependencia de conexión a Internet o de servicios en la nube.

2.1 Muestras

El sistema fue diseñado para ser utilizado por estudiantes de media con y sin fundamentos de programación, estudiantes de educación superior o profesionales en diferentes áreas de formación académica, por tal motivo las implementaciones se hicieron con estudiantes de media y profesionales en diversos campos de formación.

La selección de los participantes fue intencional y por conveniencia, criterio coherente con el alcance exploratorio del estudio. Se buscó deliberadamente contrastar los dos extremos del rango de usuarios para los que la plataforma fue diseñada: por un lado, adolescentes sin ninguna formación previa en programación, que representan el escenario de mayor exigencia para la usabilidad del sistema; por otro, adultos con formación profesional heterogénea y ajena a la IA, que permiten valorar si el diseño resiste una única sesión corta sin aprestamiento previo. El criterio de inclusión para la muestra 1 fue estar matriculado en grado undécimo en la institución educativa distrital participante y declarar no haber cursado formación en lenguajes de programación; para la muestra 2, ser profesional en ejercicio y no desempeñarse en el campo de la IA. El único criterio de exclusión aplicado fue la ausencia del consentimiento o asentimiento informado descrito en 2.6. La Tabla 2 resume las características de ambos grupos.

Tabla 2.
Caracterización de las muestras

Atributo	Muestra 1	Muestra 2
Tamaño (n)	6	18
Perfil	Estudiantes de grado undécimo, institución educativa distrital (Bogotá)	Profesionales en ejercicio de distintas áreas de formación
Rango de edad	16-19	35-60
Distribución por sexo	1 mujer, 5 hombres	15 mujeres, 3 hombres
Formación previa en programación	Ninguna declarada	7 con, 11 sin
Máximo nivel educativo	Educación media en curso	4 pregrado, 14 posgrado
Contacto previo con IA	6 uso de asistentes IA, 0 cursos de IA	15 uso de asistentes, 4 cursos de IA
Modalidad de trabajo	Individual (una AI_Card por estudiante)	Grupos de tres, con un estudiante de la muestra 1 como asesor
Dedicación	4 sesiones × 4 h y 1 sesión x1h = 17 h	1 sesión × 1 h
Rol en el estudio	Participantes y, en la sesión 5 conformadores	Participantes
Muestreo	Intencional por conveniencia	Intencional por conveniencia

Fuente: los autores.

2.1.1 Dinámica Muestra 1

Se utilizó una estrategia de aprestamiento de tres etapas, la primera de reconocimiento y familiarización con el material; la

segunda de verificación y comprobación (utilización guiada) y la tercera de experimentación autónoma. A cada estudiante se le asignó una AI_Card para el trabajo durante las tres etapas. Este trabajo se llevó a cabo en cuatro sesiones, cada una de cuatro horas. La primera sesión inició con el reconocimiento de los diferentes módulos de hardware y software de la AI_Card. A cada estudiante se le asignó una AI_Card en un kit con su correspondiente identificación (AI_Card0, AI_Card1, AI_Card2, AI_Card3, AI_Card4, AI_Card5), en la misma y segunda sesión se inició la segunda etapa (verificación y comprobación) con base en una guía desarrollada por el docente autor de la AI_Card diseñada para manejar de manera práctica cada uno de los módulos de la tarjeta (*Data, Training, Inference, iavision*) así como los diferentes componentes de hardware que la acompañan. Esta guía se encuentra de forma digital dentro de la microSD que contiene el ejecutable de la AI_Card.

En las últimas dos sesiones se desarrolló la tercera etapa (experimentación autónoma). En la tercera sesión los estudiantes trabajaron con base en retos propuestos por el autor de la AI_Card, de manera autónoma los estudiantes iniciaron con la creación de Data-sets, entrenamientos de modelos, inferencia de modelos e interactuaron con LMMs locales a través de los GPIOs de la AI_Card bajo sus propias condiciones.

En la cuarta sesión el autor de la AI_Card, el facilitador y los estudiantes diseñaron las correspondientes estrategias para que los 18 profesionales de la segunda muestra lograran interactuar durante una sesión de una hora con la AI_Card de forma asistida por el facilitador y los seis estudiantes de la muestra 1.

2.1.2 Dinámica Muestra 2

La estrategia utilizada para la segunda muestra constaba de una única etapa y una sola sesión asistida por el facilitador y los seis estudiantes de la primera muestra y con la participación de 18 profesionales. Durante la sesión de una hora los 18 profesionales se dividieron en seis grupos de tres personas al azar, cada grupo contaba con la asistencia de un estudiante de la muestra 1. La sesión consistía en que los grupos de profesionales tuvieran la oportunidad de participar activamente en la creación de data-sets, entrenamiento de redes neuronales, inferencia de modelos e interacción con LMMs locales con la AI_Card (*Data, Training, Inference, iavision*); para el control de los tiempos dentro de la actividad el facilitador dirigió la activad que fue asesorada por el estudiante de cada grupo.

Al final de esta sesión los profesionales lograron crear data-sets de entrenamiento, entrenar modelos de redes neuronales para reconocer cartas y hacer su correspondiente inferencia, así como interactuar con el LMM local gemma4:e2b a través del módulo *iavision* y observar su comportamiento en los diferentes GPIOs de la AI_Card.

2.2 Arquitectura AI_Card

La AI_Card es un sistema embebido de desarrollo propio construido sobre el microcontrolador ESP32-S3 WROOM, diseñado para ejecutar pipelines completos de visión por computadora: recolección de datos, entrenamiento de redes neuronales convolucionales (CNN) y perceptrones multicapa (MLP), inferencia en tiempo real, y control de actuadores físicos mediante GPIO. Incorpora, adicionalmente, un módulo de análisis multimodal basado en modelos de lenguaje multimodal (LMM) ejecutados localmente a través de Ollama, sin dependencia de servicios en la nube (Fig. 1).

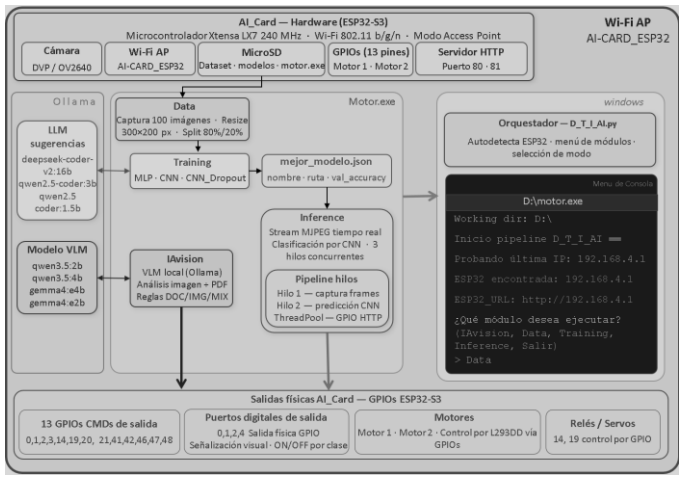

Figura 1. Arquitectura AI_Card
Fuente: Los autores.

2.2.1 Arquitectura del hardware

Microcontrolador ESP32-S3. El núcleo de la AI_Card es el ESP32-S3 de Espressif Systems, un SoC (System on Chip) de doble núcleo Xtensa LX7 a 240 MHz con soporte para instrucciones vectoriales que aceleran operaciones de redes neuronales en el propio chip. Sus características principales en el contexto de esta plataforma son:

- Conectividad Wi-Fi 802.11 b/g/n integrada, utilizada para streaming de video (MJPEG) y control bidireccional HTTP/TCP entre la tarjeta y el PC host.
- Puertos GPIO programables (GPIOs 0, 1, 2, 3, 14, 19, 20, 21, 41, 42, 46, 47, 48) mapeados a 13 categorías de clasificación.
- MicroSD (SPI) para almacenamiento de dataset, ejecutable y modelos entrenados.

Topología de red. La AI_Card opera en modo Access Point (AP), generando su propia red Wi-Fi SSID 'AI-CARD_ESP32' con contraseña configurable. El PC host se conecta como estación (STA) a esta red. Este diseño elimina la dependencia de infraestructura de red existente y garantiza latencia mínima en la transmisión del *stream* de video (Fig. 2).

```
Topología de comunicación:

  AI_Card (ESP32-S3, AP)  ⟷  PC Host (STA)

  ┌──────────────────────────────────────────┐
  │ Puerto 80  : Servidor HTTP (stream,      │
  │              captura, control GPIO)      │
  │ Puerto 81  : Control GPIO asíncrono      │
  │ /stream    : MJPEG multipart stream      │
  │ /capture   : JPEG frame único            │
  │ /gpio?state=N : Activar salida N         │
  │ /gpio_on?pin=N / gpio_off?pin=N          │
  └──────────────────────────────────────────┘
```

Figura 2. Comunicación AI_Card
Fuente: los autores

Firmware y entorno de desarrollo**.** El *firmware* se desarrolla con el IDE Arduino 1.8.6 usando el paquete esp32 de Espressif (ESP-IDF envuelto en Arduino *framework*). Las bibliotecas auxiliares necesarias para el servidor HTTP, el *streaming* MJPEG y el control de GPIOs se distribuyen preempaquetadas en la microSD y deben copiarse a la carpeta *libraries* del entorno Arduino del PC de desarrollo.

Circuito PCB y puertos. Puertos GPIO programables (GPIOs 0, 1, 2, 3, 14, 19, 20, 21, 41, 42, 46, 47, 48) mapeados a 13 categorías de clasificación con estados físicos a través de 13 LEDs tipo CMD en la AI_Card; puertos físicos de conexión RJ11: 0, 1, 2, 3 para salidas digitales, 14 y 19 para conexión de servos, M1 y M2 para conexión de motores incluido driver para motores L293DD en la AI_Card.

AI_Card-Kit. La AI_Card viene acompañada con un Kit de tarjetas-actuadores y conectores dispuestos de la siguiente forma:

a. Ocho cables rj11 macho para conexión de actuadores
b. Un cable de potencia para alimentación de motores
c. Tres tarjeta-actuador tipo LED
d. Dos tarjeta-actuador tipo servo
e. Dos tarjeta-actuador tipo motorreductor

2.2.2 Arquitectura del software

Pipeline principal: D_T_I_AI.py**.**

El software de la AI_Card se organiza en un pipeline modular orquestado por D_T_I_AI.py, que gestiona cuatro módulos interdependientes como interfaz de usuario conversacional para seleccionar módulos, aunque la lógica de ejecución es determinista una vez seleccionado el módulo (Fig. 3).

```
Módulos del pipeline D_T_I_AI:

  IAvision  →  Modo VLM multimodal (análisis de imagen + PDF)
  Data      →  Recolección y preprocesamiento del dataset
  Training  →  Entrenamiento de MLP + CNN + CNN_Dropout
  Inference →  Clasificación en tiempo real + control GPIO

Autodetección de ESP32:
  1. Carga última IP guardada en esp32_last_ip.json
  2. Escaneo paralelo de redes (ThreadPoolExecutor, 80 workers)
  3. Probe HTTP a /  /stream  /capture (puertos 80)
  4. Persiste IP encontrada para próximas sesiones
```

Figura 3. Módulos de D_T_I_AI.py
Fuente: los autores.

Módulo Data: preprocesamiento de imágenes. El módulo Data.py implementa un pipeline de preprocesamiento en dos etapas:

– Etapa 1: normalización dimensional. Todas las imágenes, independientemente de su formato original (JPEG, PNG, BMP, TIFF, WebP, GIF), se redimensionan a 300×200 píxeles mediante interpolación LANCZOS (antialias de alta calidad), con conversión forzada a espacio de color RGB. Esta normalización garantiza que el tensor de entrada a las redes neuronales tenga dimensiones fijas sin necesidad de padding o cropping.
– Etapa 2: división train/test estratificado por clase. Las imágenes adaptadas se dividen en conjuntos de entrenamiento (80 %) y prueba (20 %) por categoría (subcarpeta), asegurando representación proporcional de cada clase en ambos conjuntos. La semilla aleatoria es configurable (RANDOM_SEED) para reproducibilidad. Las imágenes se mueven físicamente (shutil.move) en lugar de copiarse, para evitar duplicación en la microSD.

Módulo Training: arquitecturas de redes neuronales**.** El módulo *Training* (Training_Config.py + D_T_I_AI.py) entrena de forma secuencial tres arquitecturas distintas sobre el mismo dataset, comparando su desempeño en el conjunto de validación para seleccionar automáticamente el mejor modelo. Las imágenes se cargan en escala de grises (1 canal) y se normalizan al rango [0, 1].

Arquitectura 1: Perceptrón Multicapa (MLP / Red Densa). La red densa aplana la imagen (300×200×1 = 60.000 valores) en un vector 1D y lo procesa mediante capas densamente conectadas (Fig. 4). Este modelo sirve como base-line para comparar con las CNN.

```
Arquitectura MLP (configurable en Training_Config.py):

  Input: (200, 300, 1)  →  Flatten: 60.000 nodos
  Dense(256, relu)  →  Dropout(0.3)
  Dense(128, relu)  →  Dropout(0.2)
  Dense(N_clases, softmax)

Parámetros (~15.6M para 13 clases).
Limitación: no captura relaciones espaciales entre píxeles.
```

Figura 4. Arquitectura red densa
Fuente: los autores

Arquitectura 2: CNN simple. Red convolucional de tres bloques Conv2D + MaxPooling2D seguidos de una capa densa. Captura jerarquías de características espaciales (bordes → texturas → formas). La entrada es una imagen en escala de grises de 300×200×1(Fig. 5).

```
Arquitectura CNN (Training_Config.py):

  Input: (200, 300, 1)
  Conv2D(32, 3×3, relu) → MaxPooling2D(2×2)
  Conv2D(64, 3×3, relu) → MaxPooling2D(2×2)
  Conv2D(128,3×3, relu) → MaxPooling2D(2×2)
  Flatten → Dense(256, relu) → Dropout(0.4)
  Dense(N_clases, softmax)

Parámetros (~2M para 13 clases, dependiendo del pooling).
```

Figura 5. Arquitectura red CNN
Fuente: los autores

Arquitectura 3: CNN_Dropout con augmentación de iluminación**.** La arquitectura más robusta incorpora cuatro bloques de doble convolución con BatchNormalization, SpatialDropout2D y regularización L2, más una capa de iluminación personalizada (LightAug) que opera únicamente durante el entrenamiento. El GlobalAveragePooling2D sustituye al Flatten para reducir el número de parámetros y mejorar la generalización (Fig. 6).

```
Capa personalizada LightAug (Keras serializable):

  class LightAug(tf.keras.layers.Layer):
      # Solo actúa en training=True
      def call(self, x, training=None):
          if training:
              x = tf.image.random_brightness(x, max_delta=0.22)
              x = tf.image.random_contrast(x, 0.75, 1.30)
              x = tf.clip_by_value(x, 0.0, 1.0)
          return x

Arquitectura CNN_Dropout:
  Input → LightAug → BatchNorm
  [Conv2D(32)×2 + BN + ReLU + MaxPool + SpatialDrop(0.05)]
  [Conv2D(64)×2 + BN + ReLU + MaxPool + SpatialDrop(0.08)]
  [Conv2D(128)×2 + BN + ReLU + MaxPool + SpatialDrop(0.12)]
  [Conv2D(256)×2 + BN + ReLU + MaxPool + SpatialDrop(0.15)]
  GlobalAveragePooling2D
  Dense(256, relu, L2=1e-4) → Dropout(0.45)
  Dense(N_clases, softmax)
```

Figura 6. Arquitectura red CNN con dropout
Fuente: los autores

Selección automática del mejor modelo. Al finalizar el entrenamiento de las tres arquitecturas, el sistema compara el máximo de val_accuracy obtenido por cada modelo y persiste el ganador en un archivo JSON (mejor_modelo.json) con el nombre, la ruta del archivo .h5 y la precisión de validación alcanzada. El módulo *Inference* carga este archivo en el arranque para usar automáticamente el mejor modelo disponible (Fig. 7).

```
# mejor_modelo.json (generado automáticamente)
{
  "nombre": "CNN_Dropout",
  "ruta": "D:/Modelo_CNN_Dropout.h5",
  "val_accuracy": 0.9723
}
```

Fuente: los autores.

Módulo *Inference*: clasificación en tiempo real. El módulo *Inference*.py implementa un sistema de clasificación concurrente con tres hilos: captura de *frames* (daemon), procesamiento/predicción y despacho de comandos GPIO. La arquitectura de hilos evita que la latencia de red bloquee la captura de video (Fig. 8).

```
Pipeline de inferencia (multihilo):

  Hilo 1: capture_frames()
    └ Abre VideoCapture(stream_url) con reconexión automática
    └ Deposita frames en frame_queue (maxsize=5, LIFO implícito)

  Hilo 2: process_frames()
    └ Extrae frame → grayscale → resize(300,200) → norm/255
    └ modelo.predict(img[np.newaxis,...])  →  argmax → label
    └ Si command != last_command: envía GPIO asíncronamente

  ThreadPoolExecutor (5 workers): send_command_async()
    └ session.get(f'{esp32_url}:81/gpio?state={N}', timeout=5)

Mapeado de 13 clases a 13 GPIOs físicos:
  clase 0 → GPIO 0  | clase 4  → GPIO 14 | clase 8  → GPIO 41
  clase 1 → GPIO 1  | clase 5  → GPIO 19 | clase 9  → GPIO 42
  clase 2 → GPIO 2  | clase 6  → GPIO 20 | clase 10 → GPIO 46
  clase 3 → GPIO 3  | clase 7  → GPIO 21 | clase 11 → GPIO 47
                                         | clase 12 → GPIO 48
```

Figura 8. Arquitectura módulo *inference*
Fuente: los autores

El preprocesamiento en inferencia convierte el frame BGR (OpenCV) a escala de grises, aplica resize a 300×200 con interpolación cúbica, normaliza a [0,1] y agrega las dimensiones de canal y batch antes de la predicción. La capa LightAug es inactiva en inferencia (training=False), lo que garantiza determinismo en las predicciones.

Módulo visión IA: análisis multimodal con LMM. El módulo *iavision*.py implementa un sistema de análisis multimodal que combina tres fuentes de información para el control de GPIOs: análisis de documentos PDF (texto extraído, procesado determinísticamente), análisis de imágenes en tiempo real por un LMM y reglas de combinación configurables por el usuario. El usuario configura reglas para cada pin GPIO usando tres tipos de condición documentos, imágenes y mixtas (Fig. 9).

```
Tipos de condición en IAvision:

  [DOC]  palabra1, palabra2   →  Activa si palabra aparece en PDF
  [IMG]  palabra1, palabra2   →  Activa si VLM menciona palabra en imagen
  [MIX AND] doc_palabras | img_palabras  →  Activa si AMBAS condiciones
  [MIX OR]  doc_palabras | img_palabras  →  Activa si CUALQUIER condición

Ejemplo de configuración:
  GPIO 0: [DOC] contrato, vigente
  GPIO 1: [IMG] persona, cara
  GPIO 2: [MIX AND] autorizado | tarjeta, QR
```

Figura 9. Arquitectura módulo *iavision*
Fuente: los autores.

El análisis de PDF es determinista: se busca la palabra clave directamente en el texto extraído (búsqueda de sub-cadena, case-insensitive), evitando la variabilidad del LMM para este tipo de consultas. El análisis de imagen usa el LMM qwen3.5:4b por defecto vía Ollama con temperatura 0 para máxima reproducibilidad, y se evalúa mediante búsqueda de palabras clave en la respuesta textual del modelo.

2.3. Decisiones de diseño y trade-offs

Escala de grises vs. color. Las redes se entrenan e infieren sobre imágenes en escala de grises (1 canal) en lugar de RGB (3 canales). Esta decisión reduce el espacio de entrada en un factor de 3, disminuye el número de parámetros y acelera el entrenamiento en PCs sin GPU dedicada. La contrapartida es pérdida de información cromática, lo que puede afectar la capacidad de discriminación en tareas donde el color es la característica principal. Para dichas aplicaciones, el arquitecto del modelo debería modificar el pipeline a 3 canales.

Redimensionado con distorsión. La función resize_stretch aplica redimensionado forzado a 300×200 sin preservar la relación de aspecto (aspect ratio). Aunque esto puede distorsionar objetos de proporciones extremas, tiene la ventaja de eliminar el padding (relleno negro o blanco) que podría convertirse en un artefacto aprendido por la red.

Entrenamiento en CPU vs. GPU. El sistema está diseñado para funcionar en PCs convencionales sin GPU dedicada. TensorFlow detecta automáticamente si hay una GPU disponible y la usa si existe. En CPU, el entrenamiento de CNN_Dropout con datasets de ~1000 imágenes y 50 épocas tarda típicamente entre 15 y 45 minutos dependiendo del hardware. La arquitectura MLP, pese a tener más parámetros

totales, puede ser más rápida de entrenar en CPU por su estructura regular.

Almacenamiento de modelos en formato .h5. Los modelos se guardan en formato HDF5 (.h5) de Keras para máxima compatibilidad, esto garantiza que pueda deserializarse correctamente al cargar el modelo en *Inference*.py, incluso cuando este módulo se ejecuta en un proceso separado o como ejecutable empaquetado (PyInstaller).

Ejecución como ejecutable empaquetado. El sistema se distribuye como motor.exe (generado con PyInstaller), lo que implica que el directorio base se resuelve. Todo el código usa la función _exe_dir() / _base_dir() para resolver rutas relativas, asegurando que tanto la versión .py como el .exe funcionen correctamente sin modificaciones.

2.4. Posibilidades de extensión e investigación

Nuevas arquitecturas de red. El archivo Training_Config.py está diseñado como punto de extensión para estudiantes e investigadores. Las funciones construir_modelo_denso(), construir_modelo_cnn() y construir_modelo_cnn2() pueden modificarse o complementarse con nuevas arquitecturas sin tocar el pipeline principal. El sistema de selección automática por val_accuracy funciona con cualquier número de modelos añadidos.

Integración de LMMs. El módulo *iavision* usa la API de Ollama, que soporta cualquier modelo compatible con dicho runtime (LLaVA, Moondream, Phi-3 Vision, Gemma3, Qwen-VL, entre otros). Cambiar el modelo requiere únicamente modificar la variable MODELO_VLM en *iavision*.py, esto permite comparar el desempeño de distintos LMMs en la tarea de control de hardware, un área de investigación activa en sistemas ciberfísicos con IA.

Data-set y métricas avanzadas. El sistema actual usa val_accuracy como única métrica de selección de modelo. Para aplicaciones de producción o investigación, se recomienda extender el pipeline con matriz de confusión, F1-score por clase, y curvas ROC, especialmente en data-sets desbalanceados. La estructura modular del código facilita añadir estas métricas al bloque de comparación de modelos en D_T_I_AI.py.

2.5 Instrumentos de recolección y análisis de datos

La evidencia del componente educativo procede de cuatro fuentes complementarias, resumidas en la Tabla 3. La primera es la observación directa estructurada del facilitador, consignada en una bitácora por sesión con categorías definidas previamente: hitos alcanzados, obstáculos técnicos, preguntas formuladas y decisiones tomadas por cada participante. La segunda son los registros automáticos que produce el propio sistema: el archivo *mejor_modelo.json* con el modelo ganador y su val_accuracy, los gráficos de precisión y pérdida por arquitectura, el número de etiquetas e imágenes de cada data-set y los tiempos de entrenamiento. La tercera son los productos elaborados por los participantes (data-sets, modelos entrenados y configuraciones de reglas del módulo *iavision*), conservados en la microSD de cada AI_Card y valorados mediante una rúbrica de cuatro criterios. La cuarta es el registro fotográfico de las sesiones.

El análisis combinó estadística descriptiva sobre los registros del sistema y análisis de contenido con categorías a priori sobre las bitácoras y los productos. La triangulación entre las tres primeras fuentes es la que permite sostener las afirmaciones de la sección 3: un hito se considera alcanzado únicamente cuando aparece simultáneamente en la bitácora del facilitador y en el artefacto correspondiente almacenado en la microSD.

Tabla 3.
Instrumentos de recolección de datos

Instrumento	Fuente	Momento	Datos que produce	Uso analítico
Bitácora de observación estructurada	Facilitador	Cada sesión	Hitos, obstáculos, preguntas, decisiones	Análisis de contenido con categorías a priori
Registros automáticos del sistema	D_T_I_AI.py / mejor_modelo.json	Cada entrenamiento	Modelo ganador, val_accuracy, épocas, tiempo, n.º etiquetas, n.º imágenes	Estadística descriptiva (Tabla 4)
Rúbrica de producto	Artefactos en microSD	Cierre de cada etapa	Calidad del data-set, del entrenamiento, del contraste por inferencia y del control de GPIO	Nivel de logro por criterio
Lista de chequeo de retos	Guía de retos	Sesiones 3 y 4 (M1); sesión única (M2)	Hitos completados / no completados	Frecuencias de logro
Cuestionario de usabilidad y satisfacción	Participantes	Fin de la implementación	Escala Likert de 5 puntos sobre facilidad de uso, claridad de la guía y utilidad percibida	Medias y desviaciones por ítem
Registro fotográfico	Sesiones	Continuo	Evidencia de la disposición de trabajo	Contextualización (Fig. 14)

Fuente: los autores

2.6 Consideraciones éticas

La implementación contó con la autorización de las directivas de la institución educativa distrital participante, otorgada mediante comunicación escrita previa al inicio de las sesiones. Los acudientes de los estudiantes de la muestra 1 firmaron consentimiento informado y los propios estudiantes otorgaron asentimiento informado, en el que se explicó el propósito del estudio, la naturaleza voluntaria de la participación, el derecho a retirarse en cualquier momento sin consecuencia académica y el uso previsto de los registros. Los profesionales de la muestra 2 firmaron consentimiento informado en los mismos términos.

Por diseño, la arquitectura del estudio minimiza el riesgo: la totalidad del procesamiento (captura de imágenes, entrenamiento, inferencia y análisis multimodal) ocurre en el equipo local y en la microSD de cada AI_Card, sin conexión a Internet y sin transferencia de datos a servicios en la nube ni a terceros. Los data-sets construidos por los participantes se compusieron de objetos, fichas, caracteres impresos y no de imágenes de personas.

3 Resultados

Los resultados se presentan separados en dos planos. El apartado 3.2 reúne los resultados técnicos, obtenidos de los registros que genera el propio sistema. El apartado 3.3 reúne los resultados educativos, es decir, los procedimientos, decisiones y dificultades documentados en las bitácoras del facilitador y en los productos conservados en la microSD de cada AI_Card, para cada una de las dos muestras.

3.1 Condiciones de la implementación

Para comprobar el desempeño de la AI_Card a cada muestra se le facilitaron 6 conjuntos, cada uno con los siguientes elementos. Un PC portátil Rizen 7 con de 16GB de RAM y 500 GB SSD de disco, sin conexión a internet, sistema operativo win11, sin Office y sin Arduino, Python o tensorflow instalados. Un documento en PDF de dos partes, la primera parte explica el manejo Hardware/software de la AI_Card, la segunda parte consta de actividades para desarrollar con la AI_Card de forma guiada dentro de la microSD de la AI_Card. Un AI_Card-kit (mencionada antes).

Cada AI_Card de cada AI_Card-Kit para el trabajo durante las sesiones contaba con una identificación única para la comunicación inalámbrica (host) con su PC: AI-CARD0, AI-CARD1, AI-CARD2, AI-CARD3, AI-CARD4, AI-CARD5; cada kit cuenta además con un soporte 3d y una prensa 3d para fijar la AI_Card a la mesa de trabajo.

3.2 Resultados técnicos del sistema

Los resultados de este apartado proceden de los archivos que el sistema genera de forma automática, *mejor_modelo.json*, los gráficos de precisión y pérdida por arquitectura y los registros de tiempo, además de las mediciones realizadas durante las sesiones.

Entrenamiento e inferencia. La tabla 4 sintetiza las configuraciones de entrenamiento ejecutadas durante la implementación La divergencia entre exactitud de validación y robustez en condiciones reales es, en sí misma, un resultado de valor didáctico. Durante la primera y segunda de manera guiada se crearon data-sets de diferentes tamaños que oscilaban entre 200 a 400 imágenes por etiqueta. Para estas sesiones se elaboraron data-sets con 2 etiquetas (tabla 4).

Tabla 4.
Configuraciones de entrenamiento y desempeño alcanzado

Caso	Etiq.	Img./etiq.	Épocas	val_acc			Ganador	Min	Comportamiento en inferencia
				MLP	CNN	CNN_Drop			
C1 (Fig. 10a)	2	200-400	14	0,50	~0,65	se pierde	CNN	25	Exactitud insuficiente; motivó un nuevo data-set
C2 (Fig. 10b)	2	54	32	~0,90	~0,84	se pierde	MLP	20	Mejora marcada; pérdida de exactitud al desplazar el objeto
C3 (Fig. 10c)	2	247	25	~0,50	~0,98	se pierde	CNN	25	Estable ante traslación, rotación y homotecia
C4 (Fig. 11)	4	< 150	11	~0,20	~0,90	se pierde	CNN	20-30	Inestable ante traslación, rotación y homotecia
C5 (Fig. 12)	4	200-300	22	~0,25	~0,98	se pierde	CNN	30-50	Estable ante traslación, rotación y homotecia
C6 (Fig. 13)	2	300	13	~0,51	~0,99	~0,99	CNN	< 20	> 90 % de precisión; diferenciación de dos cartas del juego UNO

Fuente los autores

Los primeros entrenamientos dejaron ver que la precisión en la inferencia no era apropiada, se crearon nuevos data-set sin mayores variaciones de posición. Para segundos entrenamientos se observó que en los gráficos la precisión aumentó por lo que se decidió hacer inferencias de modelos, la precisión del modelo mejoró radicalmente, pero al modificar la

posición del objeto dentro de la imagen en la AI_Card perdía la exactitud (tabla 4).

El tercer data-set elaborado contó con 247 imágenes por etiqueta contemplando traslaciones, rotaciones y homotecias, con ello se obtuvo el último grafico de entrenamiento (fig. 10) que luego fue contrastado a través de la inferencia del modelo, logrando un nivel de precisión y exactitud a prueba de las diferentes transformaciones geométricas.

Se crearon data-sets de diferentes tamaños que oscilaban entre 100 y 400 imágenes por etiqueta, crearon data-sets con 2, 3 y 4 etiquetas y se midieron los correspondientes tiempos de entrenamiento e hicieron inferencia de los modelos para contrastar los gráficos de entrenamiento (tabla 4).

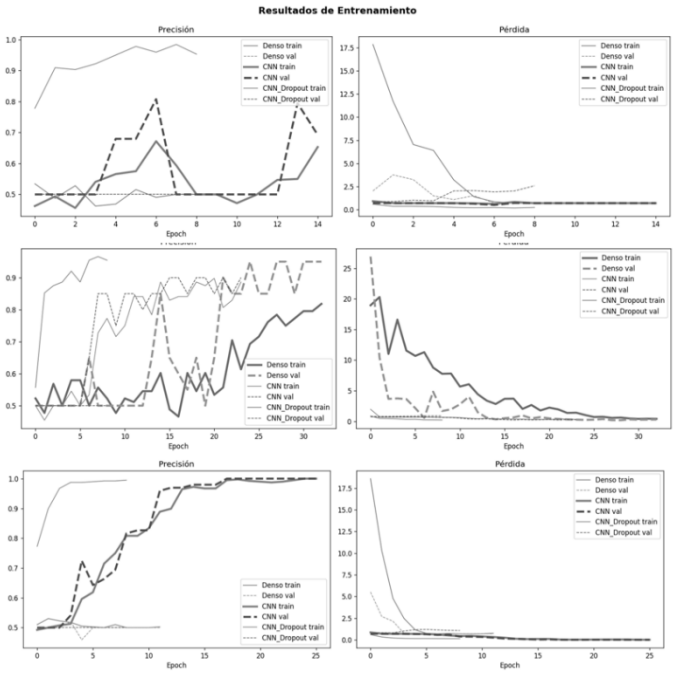

Figura 10. Gráficos de entrenamiento en tres momentos
Fuente AI_Card

Se crearon data-sets de cuatro etiquetas (Fig.11) logrando buenos niveles de precisión a pesar de contar con menos de 150 ejemplares por etiqueta.

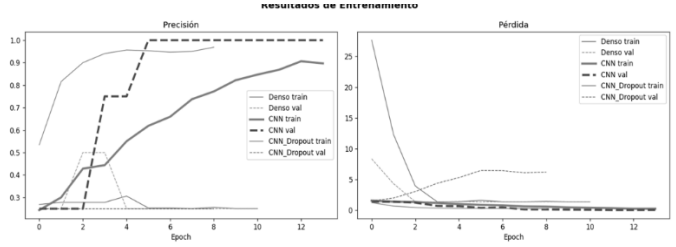

Figura 11. Gráfico de entrenamiento 4 etiquetas 150 ejemplos
Fuente: AI_Card

También se entrenaron modelos con cuatro etiquetas y conjuntos de datos entre 200 y 300 ejemplares por etiqueta (fig. 12) logrando excepcionales niveles de precisión, con tiempos de entrenamiento den entre 20 y 50 minutos.

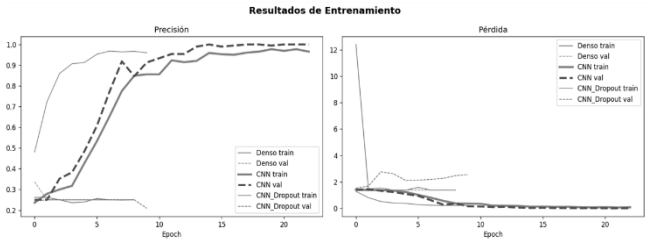

Figura 12. Gráfico de entrenamiento 4 etiquetas 300 ejemplos
Fuente: AI_Card

En la cuarta sesión en la AI_Card4 se logró crear un data-set y entrenar el modelo con tiempos de entrenamiento menores a 20 minutos y gráficos de entrenamiento arriba del 90 %, (fig. 13); las condiciones establecidas eran dos etiquetas con 300 imágenes por etiqueta que en inferencia diferenciaban dos cartas del juego UNO.

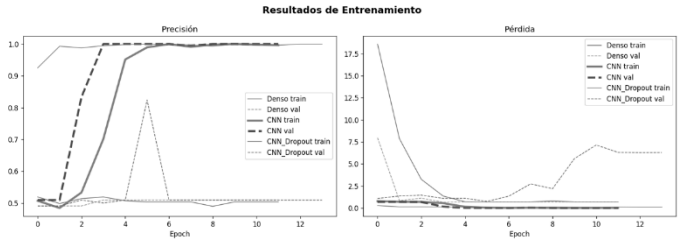

Figura 13. Gráfico entrenamiento 2 etiquetas 20 minutos
Fuente: AI_Card

3.2.2 Modelos entrenados

El acoplamiento entre la clase inferida y la salida física se verificó con los tres tipos de tarjeta-actuador del kit. La Tabla 4 recoge las configuraciones de entrenamiento asociadas a cada verificación. Después de lograr entrenamientos con niveles de precisión altos y de comprobar mediante la inferencia los modelos entrenados se procedió a conectar actuadores en las salidas físicas de la AI_Card. Iniciaron conectando los cables rj11 en las salidas 0 y 1 para evidenciar la inferencia mediante las tarjetas-actuador tipo LED, seguidamente se utilizaron las salidas 14 o 17 para conectar los cables rj11 y las tarjetas-actuador tipo servo para evidenciar la inferencia con la posición del servo, finalmente se utilizaron las salidas M1 o M2 para conectar los cables rj11 y las tarjetas-actuador tipo motor para evidenciar la inferencia de los modelos entrenados mediante la velocidad del motorreductor.

3.2.3 LMM locales

En el módulo de *iavision* se crearon diferentes prompts para los LMMs locales y evaluaron el nivel de las respuestas en función del LMM local utilizado. Para evaluar las respuestas las condiciones de *iavision* utilizadas fueron el reconocimiento de caracteres, letras, números, objetos como juguetes, cartas y fichas, así como el reconocimiento de expresiones matemáticas y ecuaciones que los estudiantes pidieron resolver a los LMMs.

La Tabla 5 recoge el contraste entre los LMMs locales evaluados sobre una misma tarea: reconocer una ecuación lineal en el conjunto de los reales presentada ante la cámara, resolverla

y activar el GPIO correspondiente al tipo de solución. El criterio de selección no fue la exactitud aislada sino la relación entre exactitud y latencia bajo la restricción operativa impuesta para el diseño de la actividad

Tabla 5.
Desempeño de LMM locales en tareas de reconocimiento y solución de ecuaciones

LMM Ollama	Tamaño	Cuantización	# pruebas	Tiempo medio	Aciertos reconoc.	Solución correcta	Activación GPIO correcta
gemma4:e2b	7,2GB	Q4_K_M	24	54s	18 de 24	19 de 24	24 de 24
gemma4:e4b	9,6GB	Q4_K_M	18	135s	15 de 18	15 de 18	18 de 18
qwen3.5:2b	2,7GB	Q8_0	18	65s	18 de 24	20 de 24	18 de 18
qwen3.5:4b	3,4GB	Q4_K_M	24	150s	24 de 24	24 de 24	24 de 24

Fuente; los autores

Después de hacerle pruebas a los diferentes modelos los estudiantes lograron ver la excelente precisión de qwen3.5 que implicaban tiempos de inferencia muy altos (tal vez debido a la calidad de los modelos *thinking*) así como los tiempos de respuesta cortos por parte del modelo gemma4:e2b con niveles de precisión aceptables; esta comparación se realizó sobre un conjunto reducido de pruebas y con prompts diseñados por los propios participantes, por lo que sus cifras describen el comportamiento observado en esta implementación y no constituyen un *benchmark* de los modelos.

3.3 Resultados educativos

Los resultados de este apartado describen lo que hicieron los participantes: qué decidieron, qué obstáculos encontraron y cómo los resolvieron. Proceden de la observación estructurada del facilitador (instrumento 1) y de los productos elaborados durante las sesiones (instrumento 3), y su alcance es el de un estudio piloto exploratorio con dos muestras intencionales, conforme a lo declarado en 2.

3.3.1 Muestra 1

Etapa 1. En esta etapa cada estudiante logró reconocer el hardware de su AI_Card, así como los elementos de software de los diferentes módulos del ejecutable en la primera sesión de trabajo.

Etapa 2. Los estudiantes entrenaron modelos neuronales predeterminados examinando los gráficos de entrenamiento (instrumento 2) y en función de estos, modificaron los data-sets para obtener mejores gráficos de entrenamiento neuronales (Fig.10) e hicieron inferencia de los modelos para contrastar los gráficos de entrenamiento.

Lograron descubrir de manera práctica que la precisión en inferencia de los modelos entrenados dependía de los gráficos de entrenamiento logrados y que estos a su vez dependían de la calidad de los data-sets y la cantidad de ejemplos dentro de estos, es por ello por lo que en la figura 10 se pueden ver tres diferentes entrenamientos para un mismo problema.

Uno de los estudiantes hizo el primer entrenamiento y vio que la precisión en la inferencia no era apropiada, así que decidió crear un nuevo data-set sin mayores variaciones de posición. Para el segundo entrenamiento observó que en los gráficos la precisión aumentó por lo que decidió hacer su correspondiente inferencia del modelo, vio que la precisión del modelo mejoró radicalmente pero que al modificar la posición del objeto dentro de la imagen en la AI_Card perdía la exactitud así que decidió crear un nuevo data-set de entrenamiento.

Los estudiantes concluyeron de manera práctica la importancia de los data-sets, los entrenamientos y la contrastación por inferencia de modelos (Fig. 10).

Etapa 3. La segunda etapa generó bastantes inquietudes en los estudiantes relacionados con los data-set y los niveles de precisión de los modelos por ello durante la tercera sesión los estudiantes de manera autónoma crearon data-sets de diferentes tamaños que oscilaban entre 100 y 400 imágenes por etiqueta, crearon data-sets con 2, 3 y 4 etiquetas.

Con la experiencia obtenida en las dos primeras sesiones algunos estudiantes crearon data-sets de cuatro etiquetas (F\ig. 11) con menos de 150 ejemplares por etiqueta con grados altos de precisión, otros decidieron entrenar modelos con cuatro etiquetas y conjuntos de datos entre 200 y 300 ejemplares por etiqueta (Fig. 12) con tiempos de entrenamiento entre 20-50 minutos.

La cuarta sesión de trabajo se centró en el diseño de las actividades que iban a desarrollar los individuos de muestra 2 con los AI_Card-kits. Para ello el autor de la AI_Card durante esta cuarta sesión exigió a los estudiantes el diseño de data-sets con entrenamientos que alcanzaran una precisión mayor 90% y con tiempos de entrenamiento de entre 10 y 20 minutos, así como el diseño de prompts y condiciones de activación de GPIOs por LMM con tiempos de respuesta de menos de 2 minutos. En esta sesión el estudiante con la AI_Card4 logró crear el data-set y entrenar el modelo con tiempos de entrenamiento menores a 20 minutos y gráficos de entrenamiento arriba del 90 %, esta estrategia fue probada por los demás estudiantes y fue validada entre los pares; las condiciones establecidas por el estudiante eran dos etiquetas que contenían 300 imágenes por etiqueta y que en inferencia diferenciaban dos cartas diferentes del juego UNO.

De igual manera se hizo con los prompts del módulo *iavision*, cada estudiante realizó pruebas con diferentes prompts, analizó tiempos de respuesta de los LMMs y probó las respuestas de los diferentes LMMs alojados en los PC.

Finalmente, el grupo llegó a una decisión conjunta del tipo de prompt que contendría el reconocimiento de una ecuación lineal en el conjunto de los reales, su solución y la activación de los GPIOs de la AI_Card en función del tipo de solución dada por el LMM en este caso gemma 4:e2b debido a las limitaciones de tiempo establecidas por el autor de la AI_Card y sopesando la precisión y los tiempos de respuesta LMM.

Al final de la cuarta sesión, los estudiantes de la muestra 1 lograron diseñar un entrenamiento con su respectiva inferencia con una duración de alrededor de 20 a 30 minutos con excelentes gráficos de entrenamiento y un diseño de prompts para interactuar con los LMMs locales a través de *iavision* con una duración de 20 minutos lo que conllevó a diseñar una actividad de una única sesión de una hora con los cuatro

módulos de la AI_Card*: Data, Training, Inference y iavision* y los diferentes elementos de Hardware que acompañan la AI_Card en su correspondiente Kit.

3.3.2 Muestra 2

A cada grupo compuesto por tres profesionales le fue asignado un AI_Card-kit para el trabajo práctico, además de contar con la asesoría permanente durante toda la sesión de un estudiante de la muestra 1, la sesión fue dirigida por el facilitador quien daba las indicaciones y marcaba los tiempos de trabajo por actividad.

Durante la sesión los profesionales de muestra dos lograron crear los data-sets de entrenamiento, hacer el entrenamiento de los modelos y lograron hacer la inferencia. Algunos grupos tuvieron la dificultad inicial al crear los data-sets con la suficiente calidad, sin embargo, esta fue solventada ampliando el margen de tiempo para dar cabida a corregir los datasets, las actividades de entrenamiento e inferencia ocurrieron en los tiempos calculados previamente y la interacción con el módulo *iavision* a través de los LMM y los GPIOs de la AI_Card sucedieron como fue planeado (Fig. 14).

Figura 14. Implementación en muestra 2
Fuente: los autores.

3.3.3 Hallazgos transversales

El contraste entre las dos muestras permite observar la misma plataforma bajo dos regímenes temporales muy distintos (dieciséis horas frente a una hora) y con dos formas de acompañamiento diferentes

Las dos estrategias utilizadas dentro de la metodología de implementación con la AI_Card mostraron la interacción de los individuos con el material en un intervalo que va desde sesiones largas hasta una única sesión de trabajo, lo que permitió observar la dinámica de la enseñanza y el aprendizaje de las redes neuronales y de la aplicación de la inteligencia artificial mediada por un sistema software-hardware en condiciones reales, evidenciando las ventajas de contar con dicho sistema para la enseñanza y el aprendizaje de las redes neuronales y la inteligencia artificial entre no especialistas en el campo.

4 Discusión

Los hallazgos de esta prueba de concepto pueden situarse frente a tres líneas de trabajo en TinyML educativo. La primera es la iniciativa que articula la especialización en TinyML de edX en torno al kit Arduino Tiny Machine Learning y a la plataforma Edge Impulse, cuyo alcance muestra que la barrera de entrada al aprendizaje automático embebido puede reducirse drásticamente [9], [10]. La segunda es la red TinyML4D, orientada de forma explícita a escalar esta formación en países en desarrollo, con actividad reciente en América Latina, incluida una sesión en Bogotá [11]. La tercera es la línea de reportes de experiencia que documenta con instrumentos estandarizados el efecto de introducir laboratorios con hardware real en cursos de computación y una preferencia marcada por el hardware físico frente a la simulación [14].

La coincidencia con esta literatura es sustantiva: los participantes de la muestra 1 manifestaron el mismo desplazamiento hacia lo tangible que reportan esos trabajos, y la afectación inicial ante la construcción del data-set observada en la muestra 2 concuerda con lo descrito en estudios de aprendizaje automático en educación básica y media [13]. La divergencia, en cambio, se ubica en dos puntos operativos. El primero es la dependencia de infraestructura externa: los flujos basados en Edge Impulse requieren conectividad, registro de cuenta y transferencia de los datos del estudiante a un servicio en la nube [12], condiciones que en instituciones oficiales colombianas no siempre pueden garantizarse y que introducen consideraciones adicionales de tratamiento de datos cuando los participantes son menores de edad. La AI_Card cierra el ciclo completo sin ninguna de esas condiciones. El segundo es el alcance del ciclo: los trabajos revisados concentran la actividad del estudiante en la recolección de datos y en el despliegue del modelo, mientras el entrenamiento ocurre en un servicio remoto. En la implementación aquí reportada, el entrenamiento sucede en el equipo del propio estudiante, la comparación entre tres arquitecturas es explícita y observable, y la inferencia se materializa en el accionamiento de actuadores conectorizados.

Un elemento adicional es la incorporación de LMM ejecutados localmente como capa de decisión sobre el hardware. Los trabajos de TinyML educativo se han concentrado en clasificadores compactos desplegados en el microcontrolador, mientras que la combinación de un clasificador entrenado por el estudiante con un LMM local que interpreta imágenes y documentos y acciona salidas físicas constituye un escenario emergente, más próximo a la literatura de sistemas ciberfísicos con inteligencia artificial generativa que a la de TinyML clásico.

Los trabajos anteriormente citados reportan tamaños de muestra sustancialmente mayores e instrumentos validados, mientras que los hallazgos preliminares de esta prueba de concepto proceden de seis estudiantes y dieciocho profesionales en una única institución. Lo que puede sostenerse a partir de la evidencia disponible es que la plataforma resulta una alternativa viable y de bajo costo para llevar el ciclo completo del aprendizaje automático al aula sin conectividad.

5 Limitaciones y trabajo futuro

Las muestras son pequeñas, intencionales y provienen de una única institución educativa, por lo que los hallazgos no son generalizables a otros contextos: lo que documentan son procedimientos, decisiones y productos observados durante la

implementación. Los instrumentos empleados fueron construidos por el equipo investigador y no cuentan con validación psicométrica. La muestra 2 trabajó durante una única hora, tiempo suficiente para completar el pipeline con acompañamiento, pero insuficiente para valorar apropiación conceptual. El trabajo futuro se enfocará en un diseño de investigación cuasiexperimental, así como en el mapeo de las actividades de la AI_Card en el marco de competencias de la UNESCO [4] de modo que la plataforma pueda articularse con las orientaciones curriculares vigentes; a nivel técnico la incorporación de métricas adicionales de evaluación de modelos y la inclusión de sistemas agenticos vía *iavisión*.

6 Conclusiones

La AI_Card es un sistema funcional de inteligencia artificial en el borde de nivel 1 que cubre el ciclo datos-entrenamiento-inferencia en un formato compacto y de bajo costo, adecuado tanto para la enseñanza de Deep learning como para hacer implementaciones prácticas a través de sus GPIOs.

La integración de LMMs locales a través de Ollama en la AI_Card abre la puerta a actualizaciones sin perder la compatibilidad que ofrecen los dispositivos con inteligencia artificial en el borde con los recursos existentes en las instituciones educativas y la no dependencia de la actualización de equipos y redes de las instituciones.

Los tres modelos entrenados en paralelo (MLP, CNN simple y CNN con regularización) permiten que la selección automática por val_accuracy elija, entre las tres arquitecturas entrenadas, la de mayor exactitud de validación para cada data-set, sin intervención manual del usuario.

La integración de LMMs locales a través de Ollama abre la puerta a aplicaciones de análisis semántico sobre imágenes y documentos con control directo de hardware, un escenario de creciente relevancia en sistemas de automatización aumentados con inteligencia artificial generativa.

La AI_Card está diseñada como punto de extensión para estudiantes e investigadores. Las funciones internas en Training_Config.py pueden modificarse o complementarse con nuevas arquitecturas sin tocar el pipeline principal y presentan una oportunidad a quienes quieren experimentar en detalle las redes neuronales y la inteligencia artificial en el borde.

Los hallazgos preliminares de esta prueba de concepto sugieren que la AI_Card constituye una alternativa viable y de bajo costo para que estudiantes de educación media y profesionales que no están en el campo de la inteligencia artificial experimenten con un dispositivo inteligencia artificial en el borde, no solo como usuarios sino como creadores. Al recolectar sus propios datos, entrenar sus propias redes y ver cómo sus modelos controlan dispositivos físicos, los participantes evidenciaron en sus productos y en sus decisiones una aproximación práctica al ciclo de un modelo. La AI_Card se perfila como una de las opciones disponibles para que las instituciones educativas colombianas incorporen dispositivos de inteligencia artificial en el borde de bajo costo en el aula.

Referencias

[1] UNESCO, “Consenso de Beijing sobre la inteligencia artificial y la educación”, 2019, pp. 27-38 https://unesdoc.unesco.org/ark:/48223/pf0000368303

[2] UNESCO, “Inteligencia artificial y educación. Guía para las personas a cargo de formular políticas” 2021. [En línea]. Disponible en: https://unesdoc.unesco.org/ark:/48223/pf0000379376

[3] UNESCO, “Marco de competencias para docentes en materia de IA”, 2025 https://unesdoc.unesco.org/ark:/48223/pf0000393813?locale=es

[4] UNESCO, “Marco de competencias para estudiantes en materia de IA”, 2025 https://unesdoc.unesco.org/ark:/48223/pf0000391105

[5] Departamento Nacional de Planeación, “Documento CONPES 4144”, 2025, https://colaboracion.dnp.gov.co/CDT/Conpes/Econ%C3%B3micos/4144.pdf

[6] D. Situnayake, & J. Plunkett, “AI at the edge: Solving real-world problems with embedded machine learning”. O’Reilly Media, 2023.

[7] P. Warden y D. Situnayake, “TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers”. O’Reilly Media, 2020.

[8] K. Sharma et al., “Revolutionizing education with Edge AI”, En “Edge Artificial Intelligence,” Cap. 26, 2025 https://doi.org/10.1002/9781394355037.ch26

[9] V. Janapa Reddi et al., "Widening access to applied machine learning with TinyML", arXiv:2106.04008, 2021. https://arxiv.org/pdf/2106.04008

[10] B. Plancher y V. Janapa Reddi, "TinyMLedu: The tiny machine learning open education initiative", en Proc. 53rd ACM Technical Symposium on Computer Science Education (SIGCSE 2022), vol. 2, 2022, p. 1159. https://doi.org/10.1145/3478432.3499093

[11] B. Plancher et al., "TinyML4D: Scaling embedded machine learning education in the developing world", Proceedings of the AAAI Symposium Series, vol. 3, n.º 1, 2024. https://ojs.aaai.org/index.php/AAAI-SS/article/view/31265/33425

[12] S. Hymel et al., "Edge Impulse: An MLOps platform for tiny machine learning", en Proceedings of Machine Learning and Systems (MLSys), 2023. https://arxiv.org/pdf/2212.03332

[13] I. T. Sanusi, S. S. Oyelere, H. Vartiainen, J. Suhonen y M. Tukiainen, "A systematic review of teaching and learning machine learning in K-12 education," Education and Information Technologies, vol. 28, n.º 5, pp. 5967–5997, 2023. https://link.springer.com/article/10.1007/s10639-022-11416-7

[14] O. Spantidi, "Theory is cool but so are microcontrollers: Computer science student reactions to Arduino TinyML", en Proc. 57th ACM Technical Symposium on Computer Science Education (SIGCSE TS 2026), 2026. https://doi.org/10.1145/3770762.3772499

[15] V. Caravella, S. Yassine y S. N. Kadry, "Test the capability of Arduino TinyML for machine learning", en EAI ROSENET 2023, Springer, 2024, pp. 161-175. https://doi.org/10.1007/978-3-031-64495-5_12

[16] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang y S. Han, "Tiny machine learning: Progress and futures", IEEE Circuits and Systems Magazine, 2023. https://arxiv.org/pdf/2403.19076

[17] Congreso de la República de Colombia, “Ley 1581 de 2012, por la cual se dictan disposiciones generales para la protección de datos personales”, 2012, https://www.funcionpublica.gov.co/eva/gestornormativo/norma_pdf.php?i=49981

[18] Ministerio de Educación Nacional de Colombia, “Orientaciones curriculares para el área de Tecnología e Informática en educación básica y media”, 2022, https://www.colombiaaprende.edu.co/sites/default/files/files_public/2022-11/Orientaciones_Curricures_Tecnologia.pdf

J.F. Cortés Garnica es Licenciado en Electrónica de la Universidad Pedagógica Nacional de Colombia (2006), Especialista en Matemática Aplicada de la Universidad Sergio Arboleda (2009), Magister en Didáctica de las Ciencias de la Universidad Autónoma de Colombia (2016) y Doctor en Educación de la Universidad de Baja California, México (2022). Se desempeña como docente investigador en la Secretaría de Educación de Bogotá y en PIXIE MINDS. ORCID: 0000-0001-6869-3745

K.D. Martínez Ospino es Técnico en sistemas informáticos del SENA (2024) y Tecnólogo en sistemas informáticos del Politécnico Santafé de Bogotá (2026) actualmente se desempeña como asistente de investigación en la Institución Educativa Manuel Cepeda Vargas IED. ORCID: 0009-0002-3960-2925