Artículo

Interfaz gráfica para generación de claves en el acceso a recintos.

J. Jesús Cano Morales [1,*], Ricardo Camacho Rivera [2] y Benigno Muñoz Barrón [3]

[1] Tecnológico Nacional de México, campus ITESHU, departamento de Mecatrónica, México
[2] Tecnológico Nacional de México, campus ITESHU, departamento de Mecatrónica, México
[3] Tecnológico Nacional de México, campus ITESHU, departamento de Mecatrónica, México
* Correspondencia: jjcano@iteshu.edu.mx.

Resumen: La creciente necesidad de sistemas seguros de autentificación ha impulsado el desarrollo de tecnologías que integren inteligencia artificial y hardware embebido. En este trabajo se presenta una Interfaz Gráfica de Usuario (GUI) para el reconocimiento de dígitos manuscritos utilizando Redes Neuronales Convolucionales (CNN) optimizadas con el operador T-Max-Avg. La GUI, implementada en Python, permite configurar parámetros como coeficiente de aprendizaje, épocas y umbral de agrupamiento. El sistema fue entrenado con la base de datos MNIST, alcanzando una precisión del 99.35 % en entrenamiento y 98.97 % en prueba. Las predicciones manuales realizadas por usuarios superaron el 99.2 % de precisión promedio, con comunicación directa al sistema Arduino para activar un mecanismo físico de acceso. Estos resultados demuestran la viabilidad del enfoque propuesto para su aplicación en contextos reales. Se sugiere que investigaciones futuras incluyan módulos adaptativos que incrementen la flexibilidad y seguridad del sistema.

Keywords: *aprendizaje profundo; reconocimiento de dígitos; clave de acceso.*

Citar este trabajo: Cano, J.; Camacho, R.; Muñoz, B. Interfaz gráfica para generación de claves en el acceso a recintos. *REIA* **2025**, *9*, 3.

Recibido: 29/11/2025
Aceptado: 04/12/2025
Publicado: 27/02/2026

1. Introducción

Desde una perspectiva biológica, los seres humanos poseen la capacidad innata de reconocer rostros, imágenes y caracteres escritos con gran rapidez y precisión, incluso bajo condiciones no ideales de iluminación o visibilidad. Esta habilidad, que no requiere un entrenamiento intensivo ni una preparación previa específica, resulta de complejos procesos neurobiológicos que han sido refinados por la evolución. En cambio, los sistemas computacionales, a pesar de contar con velocidades de procesamiento de datos mucho mayores, requieren técnicas avanzadas de aprendizaje automático (*machine learning*) y aprendizaje profundo (*deep learning*) para alcanzar niveles de desempeño similares en tareas perceptuales [1].

El desarrollo de estos sistemas inteligentes ha permitido que las máquinas no solo adquieran conocimiento a partir de datos históricos, sino que también sean capaces de tomar decisiones, clasificar información y reconocer patrones complejos. Para lograr este tipo de funcionalidad, se utilizan métodos de aprendizaje que simulan —en un sentido computacional— los mecanismos adaptativos del aprendizaje humano. De esta forma, mediante la exposición a grandes volúmenes de datos, las máquinas pueden ajustar sus parámetros internos con el fin de mejorar la precisión en la predicción de comportamientos o la categorización de estímulos previamente desconocidos [2].

El aprendizaje automático se divide generalmente en tres enfoques principales: aprendizaje supervisado (*supervised learning*), aprendizaje no supervisado (*unsupervised learning*) y aprendizaje por refuerzo (*reinforcement learning*). El primero de ellos, y el más utilizado en aplicaciones prácticas de clasificación de imágenes, implica entrenar un modelo con datos previamente etiquetados. Esto permite que, una vez completado el entrenamiento, el sistema pueda reconocer nuevas entradas y asignarles una categoría con

cierto grado de confianza. El aprendizaje no supervisado, por otro lado, se orienta a descubrir patrones ocultos en datos no etiquetados, mientras que el aprendizaje por refuerzo se fundamenta en un mecanismo de recompensas y penalizaciones, muy útil en entornos de toma de decisiones en tiempo real [3].

En el caso del reconocimiento de dígitos manuscritos, el enfoque supervisado es particularmente adecuado, ya que se dispone de bases de datos extensamente etiquetadas, como MNIST (*Modified National Institute of Standards and Technology*), que permiten evaluar de manera sistemática el desempeño de diferentes modelos de clasificación. Esta base contiene 70,000 imágenes en escala de grises de 28×28 píxeles que representan los dígitos del 0 al 9, divididas en 60,000 muestras para entrenamiento y 10,000 para prueba [4]. Además, extensiones como MNIST-MIX han incorporado variantes culturales y lingüísticas, enriqueciendo los datos con muestras escritas en diferentes alfabetos y estilos gráficos, lo cual favorece el desarrollo de modelos con mayor capacidad de generalización.

Aunque los clasificadores lineales han sido tradicionalmente utilizados en tareas de reconocimiento, su efectividad es limitada cuando se enfrentan a problemas no linealmente separables, lo cual es frecuente en datos reales. Esto ha dado paso a una amplia adopción de modelos más complejos, como las Máquinas de Soporte Vectorial (SVM), los Perceptrones Multicapa (MLP) y, en particular, las Redes Neuronales Convolucionales (CNN), las cuales han demostrado un rendimiento sobresaliente en tareas de visión por computadora [5].

Las CNN están diseñadas específicamente para explotar la estructura espacial de las imágenes. Utilizan filtros convolucionales que se deslizan sobre la imagen de entrada para detectar patrones locales, como bordes, curvas o esquinas. Posteriormente, aplican capas de agrupamiento (*pooling*) que reducen la dimensionalidad de los datos, lo cual permite acelerar el entrenamiento y reducir la sensibilidad a pequeñas variaciones en la entrada. Esta arquitectura jerárquica permite que las CNN capten progresivamente características más complejas de las imágenes, desde patrones básicos hasta estructuras semánticas más abstractas [6].

No obstante, la implementación de redes neuronales profundas conlleva importantes desafíos computacionales, especialmente en contextos donde se requiere un alto rendimiento en dispositivos con recursos limitados, como sistemas embebidos o plataformas de bajo costo. Para atender esta necesidad, se han propuesto técnicas de compactación y optimización que permiten reducir significativamente la cantidad de parámetros y operaciones requeridas sin sacrificar la precisión del modelo. Un ejemplo notable es la técnica BUnit-Net (*Basic Unit Network*), basada en el apilamiento de unidades híbridas sobre perceptrones multicapa. Esta arquitectura logra reducir en un 97 % las operaciones en punto flotante (FLOP) y en un 96 % la cantidad de parámetros, manteniendo una pérdida de precisión inferior al 1 % [7].

Además, se han desarrollado mejoras en las operaciones de agrupamiento mediante el uso del operador T-Max-Avg, que combina las ventajas de los métodos tradicionales de *max pooling* y *average pooling*. Este enfoque ha demostrado una mejora de hasta el 2.3 % en la precisión global en comparación con técnicas estándar [8], al permitir una mejor retención de información espacial y atenuar el impacto de valores atípicos. A su vez, la hibridación de modelos CNN con algoritmos evolutivos, como los algoritmos genéticos o los algoritmos de enjambre (PSO, ABC), ha permitido optimizar hiperparámetros como la tasa de aprendizaje, el número de capas, el tamaño del *kernel* y el tamaño del lote de entrenamiento, alcanzando precisiones superiores al 99.9 % en el conjunto MNIST [9].

Por otro lado, arquitecturas más recientes como las redes generativas antagónicas (GAN) y los auto-codificadores (AE) también han demostrado su potencial en tareas de reconocimiento de escritura, especialmente cuando se dispone de pocos datos etiquetados. Estas redes, capaces de aprender representaciones latentes eficaces, han sido aplicadas exitosamente en áreas como la medicina, el monitoreo energético y la seguridad biométrica [10].

En el ámbito de la interacción usuario-máquina, las GUI se han convertido en una herramienta clave para la visualización, control e interpretación de modelos de inteligencia artificial. Las GUI permiten a usuarios no expertos interactuar con sistemas complejos de manera intuitiva y efectiva. Diversos estudios han abordado la integración de interfaces GUI con modelos de reconocimiento basados en CNN, logrando altos niveles de precisión. Por ejemplo, se ha desarrollado un generador automático de interfaces gráficas que, mediante modelos CNN, logra una clasificación del 94 % de los elementos visuales [11]. Asimismo, sistemas embebidos en aplicaciones médicas han logrado un desempeño de hasta 98.9 % en tareas de monitoreo visual mediante reconocimiento óptico de dígitos en pantallas [12].

En conjunto, el reconocimiento de caracteres manuscritos no se limita a un ejercicio académico de clasificación de imágenes, sino que constituye una herramienta poderosa con aplicaciones reales y crecientes. Su integración en interfaces inteligentes permite diseñar soluciones adaptativas y personalizadas, con alto grado de automatización y seguridad [13]

En este contexto, el uso de Python como lenguaje de programación principal resulta altamente conveniente. Bibliotecas como TensorFlow y Keras permiten diseñar, entrenar e implementar modelos de redes neuronales convolucionales; mientras que herramientas como tkinter, PIL y pyserial facilitan el diseño de interfaces gráficas, el procesamiento de imágenes y la comunicación con dispositivos externos. Este ecosistema de desarrollo abierto y multiplataforma ha permitido democratizar el acceso a la inteligencia artificial aplicada, permitiendo su adopción en ámbitos educativos, industriales y comunitarios [14].

La evolución del hardware también ha contribuido a esta tendencia. Dispositivos como Computadora Personal, Arduino, Raspberry Pi, otros microcontroladores de bajo costo, junto a diferentes sensores como Lápiz Óptico, Dispositivo Táctil utilizando el dedo para escribir, permiten implementar soluciones completas que integran la entrada gráfica, el procesamiento en tiempo real y la activación de actuadores físicos en respuesta a eventos reconocidos [15].

Estos desarrollos han sido aplicados a contextos industriales y de seguridad, como el control de accesos, la verificación de identidad, y la autentificación de usuarios a través del reconocimiento de patrones biométricos. En particular, el reconocimiento de dígitos manuscritos como mecanismo de validación de contraseñas ofrece una alternativa más segura y flexible frente a métodos convencionales [16].

Por otra parte, se han realizado experimentos exitosos de escritura aérea utilizando sistemas como *AirWrite*, donde el usuario realiza trazos en el aire y estos son reconocidos mediante sensores de ondas milimétricas, expandiendo así las posibilidades de interacción sin contacto [17]. Este tipo de innovación representa una extensión natural del reconocimiento clásico en pantalla y puede ser adaptado para sistemas de autentificación en entornos donde la higiene o la accesibilidad sean prioritarios.

Por lo tanto, el objetivo del presente trabajo es desarrollar una GUI que permita el reconocimiento de dígitos manuscritos, implementada mediante una CNN optimizada con el operador T-Max-Avg, programada en entorno Python, con retroalimentación visual y comunicación serial hacia un sistema de control Arduino, orientada a la generación de claves de acceso personalizadas para recintos académicos en el Instituto Tecnológico Superior de Huichapan.

2. Materiales y Métodos

2.1. Componentes físicos

En esta subsección, se presentan las características de los elementos físicos que interactúan con la Interfaz Gráfica de Usuario desarrollada para este sistema.

También, es fundamental comprender la sinergia y la utilidad del equipo físico incluido en el proyecto, ya que esto permitirá manipularlo adecuadamente a través de la GUI, que es el objeto de estudio.

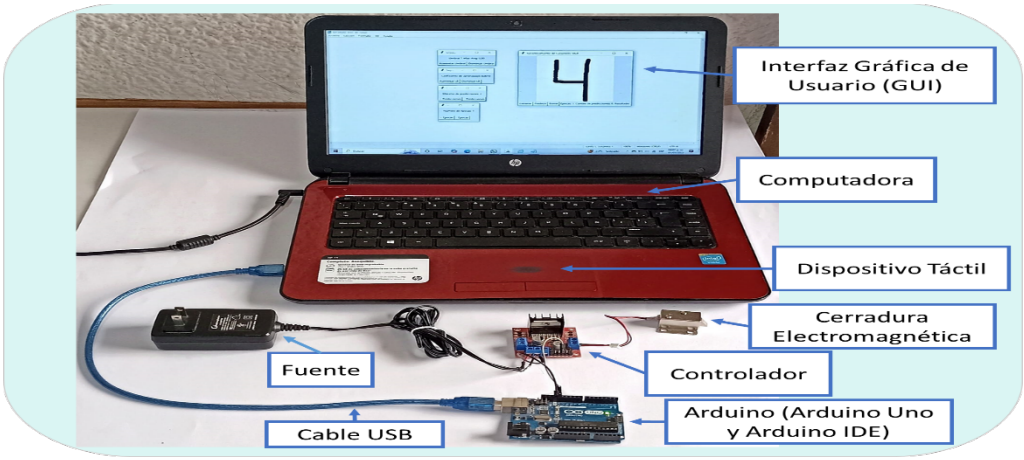

Figura 1. Interfaz de usuario en comunicación con los elementos físicos.

El sistema de control de acceso completo se puede observar funcionando, de manera real, en la Figura 1.

2.1.1. Fuente

Se utiliza para alimentar la etapa de potencia, representada por el Controlador, proporcionando los niveles adecuados de tensión y corriente eléctrica necesarios para activar la Cerradura Electromagnética. Las características se muestran en la Tabla 1.

Tabla 1. Especificaciones de la Fuente.

Modelo	Entrada	Salida
SFF1200150A1BA	100-240 VAC, 0.4 A, 50/60 Hz	12 V, 1.5 A

Es importante destacar que la Fuente es un componente electrónico imprescindible para el funcionamiento del proceso, aunque no depende directamente de la GUI.

2.1.2. Cerradura Electromagnética

Es un elemento esencial de actuación con naturaleza electromagnética, ubicado al final de la secuencia general. Recibe una señal eléctrica desde el Controlador, el cual previamente amplifica dicha señal. Esta tarjeta, a su vez, procesa una señal generada de manera digital por la plataforma Arduino, que también valida la clave de acceso correcta obtenida a través de la GUI. Las especificaciones se observan en la Tabla 2.

Tabla 2. Propiedades de la Cerradura Electromagnética.

Marca y Material	Dimensiones	Consumo
Sonew, Hierro	10 x 10 x 10 mm.	12 V, 0.4 A

La característica principal de este actuador es la expansión y contracción lineal de un solenoide, el cual integra un elemento metálico de hierro en forma prismática que actúa como cerrojo. Este mecanismo está diseñado para ser instalado en la puerta de un cubículo de profesor.

2.1.3. Controlador

El módulo está basado en el L298N, un circuito integrado que incluye un controlador de motores con un puente H doble [18]. Las particularidades se pueden ver en la Tabla 3.

Tabla 3. Características del módulo Controlador.

Nombre /Función	Modelo/Identificador	Magnitud/Cantidad
Puente completo L298N (entradas-salidas-regulados)	L298N	4-4-5 V
Voltaje de potencia	_	5-35 V
Corriente	_	0-2 A
Potencia	_	0-20 W
Regulador de voltaje (entrada-salida)	LM7805	12-5 V
Diodos de protección	1N4007	8 diodos

Su función es recibir una señal de baja magnitud proveniente del Arduino como entrada y generar como salida una señal amplificada que se dirige a la carga; en este caso, una Cerradura Electromagnética.

2.1.4. Arduino

Es una plataforma de desarrollo de proyectos ampliamente utilizada por estudiantes e investigadores, incluso por aquellos que no cuentan con conocimientos avanzados en programación y electrónica. En la Tabla 4, es posible ver algunas características de la tarjeta Arduino Uno que es parte de este entorno.

Tabla 4. Características de Arduino Uno.

Nombre/Función	Modelo/Identificador	Magnitud/Cantidad
Microcontrolador	ATmega328P	1
Resonador cerámico	CSTCE16M0V53-R0	16 MHz
Pines digitales de entrada/salida	DIGITAL, PWM	14, 6
Entradas analógicas	ANALOG_IN	6
Voltaje entrada/operación	Jack DC, USB (Cable A-B)	7-12 V, 5 V
Memorias	FLASH, RAM, EEPROM	32 KB, 2 KB, 1 KB
Comunicación USB a Serial	ATmega16U2	5 V
Regulador de voltaje (entrada/salida)	NCP1117	7-12 V, 5 V
Botón de reinicio	RST	1

Es importante mencionar que la comunidad de Arduino está en crecimiento y cualquiera de las actualizaciones se puede consultar en [19].

El Entorno Integrado de Desarrollo Arduino IDE (*Integrated Development Environment*) [20], es un programa que permite la comunicación serial, a través de un cable USB, entre la Computadora y el Arduino Uno. En el presente trabajo, la Computadora incluye una GUI que se encarga de generar y enviar los dígitos necesarios para procesar la clave de acceso. Por lo tanto, las características de la plataforma Arduino, son las siguientes:

- **Bajo costo**. Las placas Arduino son conocidas por ser más económicas que las de otras plataformas, lo que las hace atractivas para estudiantes y aficionados con presupuestos limitados.
- **Multiplataforma**. Arduino IDE funciona en Windows, Macintosh OSX y Linux.

- **Programación amigable**. Arduino IDE está basado en el entorno de programación *Processing*, por lo que, si estudiantes y profesores aprenden a programar en este último ámbito, les será sencillo comprender su uso.
- **Programas de código abierto y extensible**. Se publica como herramientas de código abierto, para ser mejorado por algún programador. El lenguaje se puede extender con librerías de C++.
- **Dispositivos extensibles**. Los planos se publican con licencia *Creative Commons*, por lo que se pueden hacer versiones propias o mejoradas.

2.1.5. Computadora

Se utiliza una portátil HP 14-R018LA para el presente proyecto. Es importante destacar que la Computadora contiene la GUI que reconoce los dígitos escritos a mano a través del Dispositivo Táctil como entrada física. En la Tabla 5, es posible ver algunas características de la Computadora.

Tabla 5. Especificaciones de la Computadora.

Nombre/Función	Modelo/Identificador	Magnitud/Cantidad
Computadora	HP 14-R018LA	14''
Procesador	Intel® Core™ i3-4005U CPU	1.70 GHz
Memorias	RAM, Disco duro	8.00 GB, 750 GB
Dispositivo Táctil	_	1
Teclado y Pantalla (cada una)	_	1
Sistema Operativo	Windows 10	64 bits

Esta interfaz genera una clave de acceso que se envía al Arduino, que procesa la información recibida y genera una señal digital hacia el Controlador. Finalmente, tras ser amplificada, esta señal activa la Cerradura Electromagnética.

2.2. Componentes virtuales

2.2.1. Base de Datos MNIST

Es importante mencionar que la base de datos MNIST se ha utilizado ampliamente a lo largo del tiempo para el reconocimiento de dígitos escritos a mano.

Este conjunto de datos ha sido fundamental para validar algoritmos de vanguardia en clasificación de imágenes, mediante CNN y técnicas de aprendizaje profundo optimizadas [21].

2.2.2. Redes Neuronales Convolucionales (CNN)

Las CNN son una de las arquitecturas de redes neuronales más utilizadas debido a su alto rendimiento en el reconocimiento de imágenes y patrones.

Como se mencionó anteriormente, estas redes se pueden entrenar con la base de datos MNIST para aprender a clasificar dígitos manuscritos.

Una vez entrenado el modelo, se compara su salida con los dígitos escritos en el Dispositivo Táctil, los cuales se visualizan en tiempo real mediante una ventana en la GUI.

Entonces, se prepara una CNN para dos capas convolucionales [22]. En la primera capa convolucional, los datos de la entrada de 28 x 28 se convolucionan a través de 32 filtros de tamaño 3 x 3 para la extracción de características espaciales.

Después, se procesa la primera capa de agrupación de submuestreo de 2 x 2, usando T-Max-Avg para mejorar la precisión tanto local como global, además de reducir la sensibilidad al ruido [23-24].

De manera análoga, en la segunda capa convolucional se filtra la matriz de salida de la primera capa con 64 filtros de tamaño 3 x 3.

Nuevamente, para reducir la dimensión del eje temporal, se realiza un submuestreo con una capa de agrupación T-Max-Avg de 2 x 2.

En este momento, la segunda capa se convierte en un vector de tamaño 1600, que son las entradas a una red neuronal totalmente conectada. Finalmente, se aplica el clasificador softmax para 10 dígitos, como se muestra en la Figura 2 [25].

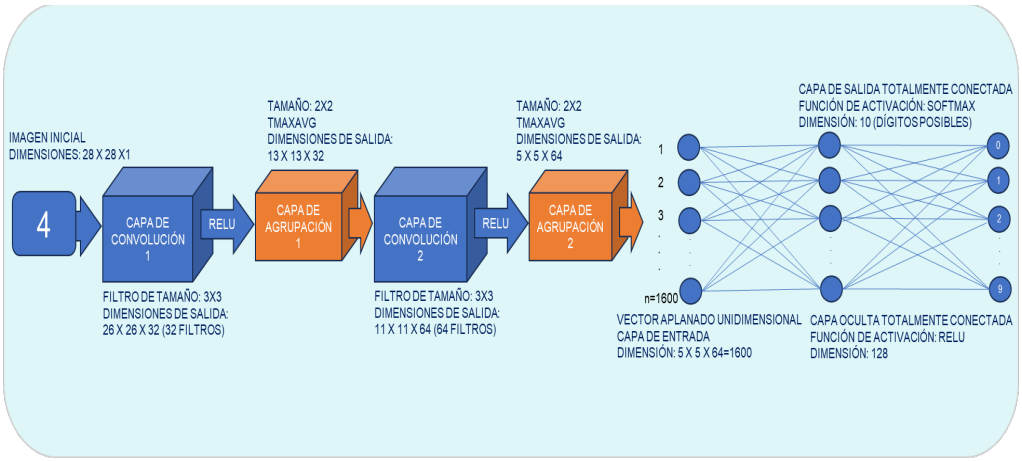

Figura 2. Estructura de la CNN.

2.2.3. Interfaz Gráfica de Usuario (GUI)

La GUI, está diseñada de manera modular [26], con canvas principal, botones de control, ventanas secundarias o sub ventanas: Predicciones, coeficiente de aprendizaje (LR), épocas, métricas de entrenamiento, gráficas de desempeño, métricas por dígito. Es importante mencionar que ha sido realizada en el entorno de Python y sus librerías se pueden explicar de la siguiente manera, para el caso de las relacionadas a las redes neuronales y procesamiento de datos:

- **TensorFlow y Keras**: Red neuronal CNN con capa personalizada [27-28].
- **NumPy:** Procesamiento numérico (matrices, métricas).

Para el caso de las librerías relacionadas con la GUI, procesamiento de imagen y comunicación de datos:

- **tkinter**: Interfaz gráfica con botones, etiquetas y canvas de dibujo.
- **PIL:** Dibujar, redimensionar e invertir imágenes.

Para la comunicación con Arduino, se declara la librería:

- **serial**: Enviar resultados de predicción al Arduino.

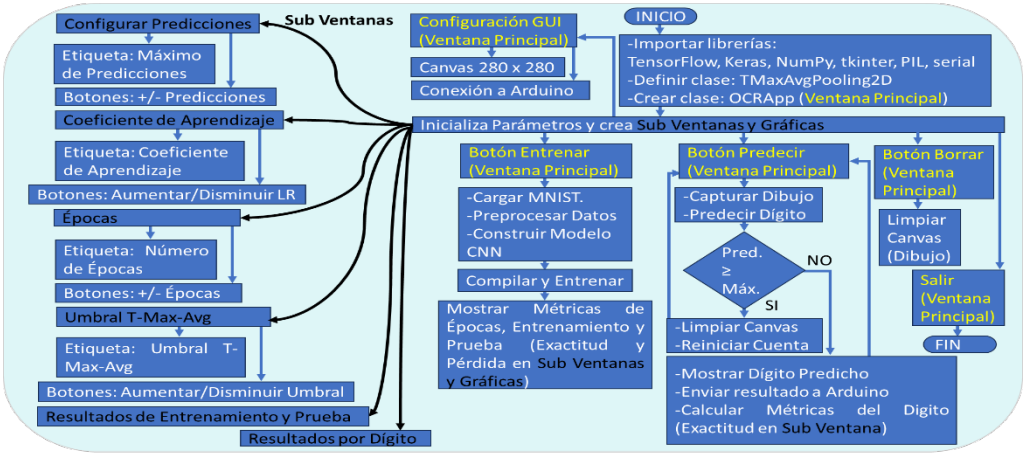

Figura 3. Diagrama de flujo estructural de la GUI.

En el diagrama de flujo de la Figura 3, se observan las librerías utilizadas, clases creadas, formación de elementos de la ventana principal (canvas, botones y etiquetas

informativas). También, es importante comentar la generación de diferentes ventanas secundarias que permiten configurar parámetros y mostrar resultados. Posteriormente, ventana principal y ventanas secundarias, se describen de manera detallada.

2.2.3.1. Ventana Principal (OCRApp)

- **Título**: "Reconocimiento de Caracteres OCR".
- **Componentes**:
    - **Canvas (Lienzo)**:
        - Tamaño: 280 x 280 píxeles.
        - Color de fondo: blanco.
        - Permite dibujar caracteres con el ratón (simula un dígito escrito a mano).
    - **Botones Inferiores**:
        - **Entrenar (train_button)**: Inicia el entrenamiento del modelo.
        - **Predecir (predict_button)**: Realiza una predicción del dígito dibujado.
        - **Borrar (clear_button)**: Limpia el lienzo y reinicia la imagen.
    - **Etiquetas Informativas**:
        - **Épocas (epochs_label)**: Visualiza el número de épocas configuradas para el entrenamiento.
        - **Resultado (result_label)**: Muestra el dígito predicho por el modelo.
        - **Conteo de predicciones (count_label):** Indica cuántas predicciones se han realizado antes de reiniciar.

2.2.3.2. Ventanas Secundarias

a. Configuración de Predicciones (predictions_window)

- **Título**: "Configuración de Predicciones".
- **Componentes**:
    - **Etiqueta**: Muestra el número máximo de predicciones permitidas y actualizadas, antes de borrar el lienzo.
    - **Botones**:
        - **+ Predicciones (increase_predictions_button)**: Aumenta la cantidad de predicciones reprogramables (hasta cuatro, pero se puede ampliar).
        - **- Predicciones (decrease_predictions_button)**: Disminuye hasta el mínimo (uno), la cifra de predicciones modificables.

b. Coeficiente de Aprendizaje (lr_window)

- **Título**: "Seguimiento del Coeficiente de Aprendizaje".
- **Componentes**:
    - **Etiqueta**: Muestra el valor actual del coeficiente de aprendizaje.
    - **Botones**:

- **Aumentar LR (increase_lr_button):** Incrementa el coeficiente en intervalos de 0.0001 (se recomienda hasta 0.001, pero se puede extender).
- **Disminuir LR (decrease_lr_button):** Reduce el coeficiente en pausas de 0.0001 (hasta el 0.0001).

c. Épocas (epochs_window)

- **Título**: "Seguimiento de Épocas".
- **Componentes**:
  - **Etiqueta**: Muestra el número de épocas configuradas para el entrenamiento.
  - **Botones**:
    - **+ Épocas (increase_epochs_button):** Aumenta el número de épocas, en incrementos de uno, hasta cinco, pero se puede ampliar.
    - **- Épocas (decrease_epochs_button):** Disminuye el número de épocas, en decrementos de uno, hasta uno como mínimo.

d. Umbral T-Max-Avg (threshold_window)

- **Título**: "Seguimiento del Umbral T-Max-Avg".
- **Componentes**:
  - **Etiqueta**: Muestra el valor actual del umbral.
  - **Botones**:
    - **Aumentar Umbral (increase_threshold_button):** Incrementa el umbral en lapsos de 0.05, hasta 0.80, pero se puede aumentar hasta uno.
    - **Disminuir Umbral (decrease_threshold_button):** Reduce el umbral en pasos de 0.05, hasta 0.05 como mínimo.

e. Resultados del Entrenamiento (train_results_window)

- **Título**: "Resultados del Entrenamiento".
- **Componentes**:
  - **Etiqueta**: Muestra la Precisión (0% al 100%) y la Pérdida del entrenamiento, que debería ser cercano a cero. El modelado, se compila basado en entropía cruzada categórica, utilizando el optimizador Adam (*Adaptive Moment Estimation*). También, se despliega Gráfica con resultados.

f. Resultados de Prueba (test_results_window)

- **Título**: "Resultados de Prueba".
- **Componentes**:
  - **Etiqueta**: Muestra la Precisión (0% al 100%) y la Pérdida del conjunto de prueba, que debería ser cercano a cero. También, se despliega Gráfica con resultados.

g. Resultados por Dígito (digit_window)

- **Título**: "Resultados para el dígito [X]".
- **Componentes**:
  - **Etiqueta**: Muestra la precisión (0% al 100%), específica para el dígito predicho.

Esta estructura permite una interacción intuitiva para el reconocimiento de dígitos escritos a mano, con opciones de personalización y retroalimentación detallada, en la Figura 4 se visualiza la forma real de la GUI, donde se puede observar la parte frontal principal.

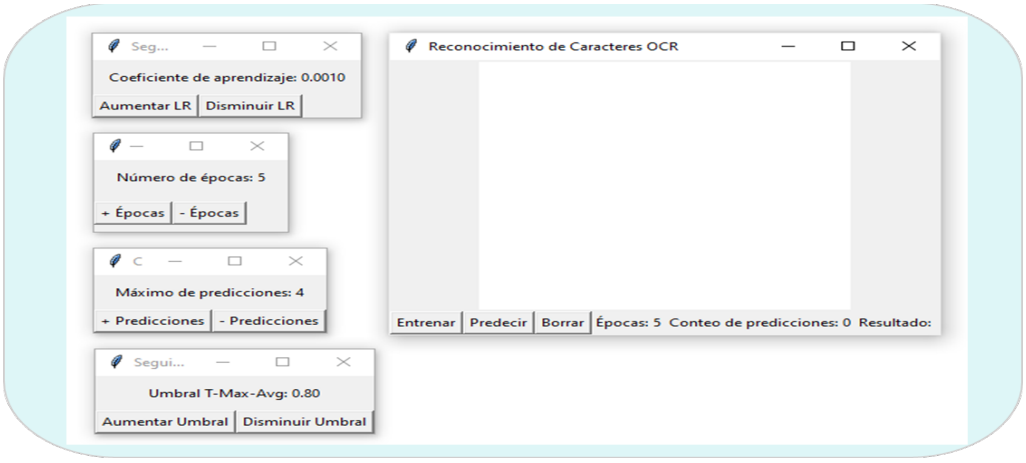

Figura 4. Vista frontal principal de la Interfaz Gráfica.

2.2.3.3. Instrucciones Generales del uso de la GUI en el control de accesos

En primer lugar, el usuario puede ajustar parámetros como épocas, coeficiente de aprendizaje, umbral T-Max-Avg, máximo de predicciones, mediante las ventanas secundarias y botones respectivos. Después, al hacer clic en el botón "Entrenar", se entrena el modelo y se muestran los resultados en ventanas emergentes (incluyendo las gráficas) de precisión, pérdida (de entrenamiento y prueba o validación). Se utiliza MNIST, con modelo CNN en versión T-Max-Avg, como se observa en la Figura 5. Posteriormente, se escribe un dígito con el dedo en el lienzo principal como imagen de entrada (a través del Dispositivo Táctil de la Computadora), cuidando de no salir del área pertinente (se puede usar el botón "Borrar", en caso de equivocarse).

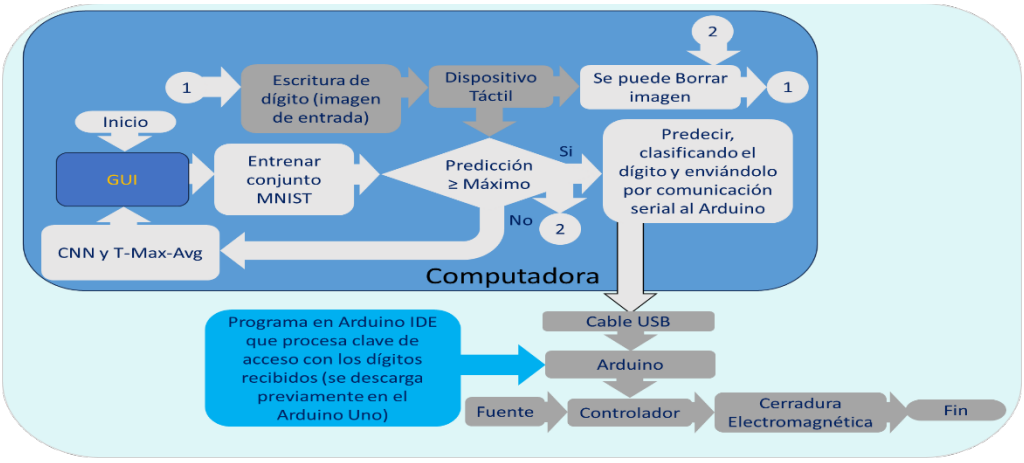

Figura 5. Diagrama de la secuencia completa del control de accesos con la GUI.

Luego, al hacer clic en el botón "Predecir", el modelo clasifica el dígito (lo reconoce, genera) y muestra el resultado en la etiqueta result_label (valor del dígito), se envía la

predicción a Arduino a través del puerto serial de comunicación COMX configurado (a través del Cable USB). Se puede hacer uso también del botón "Borrar" para escribir otro dígito. Entonces, se abre una ventana que muestra las métricas específicas para cada dígito predicho (precisión). Cuando se alcanza el máximo de predicciones configurado, el lienzo se borra automáticamente. Si se recibe la clave de acceso adecuada en la plataforma de Arduino, se envía un bit discreto hacia el elemento de amplificación (Controlador alimentado por una Fuente de poder), que a su vez habilita la Cerradura Electromagnética y el control de accesos.

Disponibilidad de datos y código: La base de datos MNIST se encuentra disponible públicamente en http://yann.lecun.com/exdb/mnist. El código fuente completo de la GUI, el modelo CNN, y los scripts de entrenamiento y comunicación serial han sido depositados en el repositorio público https://github.com/jjcano-coder/gui-cnn-claves.git (acceso consultado el 01 de agosto de 2025).

Consideraciones éticas: Este estudio no involucra investigación con sujetos humanos ni animales, por lo que no requiere aprobación por parte de un comité de ética.

3. Resultados

Con el objetivo de validar la funcionalidad del sistema propuesto, se llevaron a cabo tres etapas de evaluación: una inicial, una intermedia y una final. En cada una se evaluó el rendimiento del modelo CNN con T-Max-Avg integrado en la GUI, utilizando la base de datos MNIST y pruebas de predicción reales mediante la interfaz táctil conectada al sistema Arduino.

3.1. Evaluación inicial del modelo CNN

Durante la primera prueba, el modelo fue entrenado con 60,000 imágenes de la base de datos MNIST y validado con 10,000 muestras.

Tabla 6. Resultados de la evaluación inicial.

Tipo de Medición	Precisión (%) / Pérdida	Promedio de Precisión (%)
Entrenamiento	99.35/0.0199	_
Prueba	98.97/0.0315	_
Predicción de Dígitos Real (1, 2, 3, 4)	99.21, 99.22, 99.60, 98.88	99.23

Se alcanzó una precisión de entrenamiento de 99.35 % con una pérdida de 0.0199, y una precisión de prueba de 98.97 % con pérdida de 0.0315.

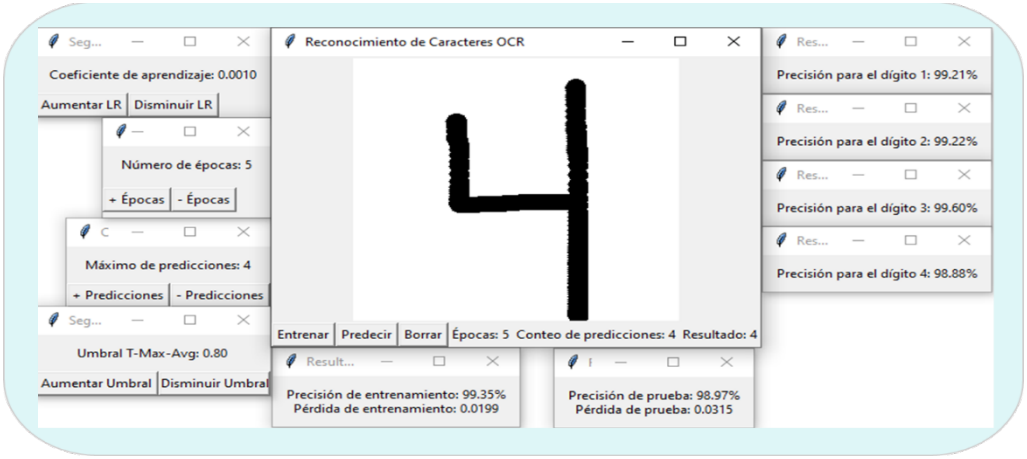

Figura 6. Evaluación inicial de funcionalidad de la interfaz gráfica.

Además, se realizaron predicciones manuales de los dígitos 1, 2, 3 y 4, obteniéndose valores de precisión superiores al 98.8 %, como se muestra en la Tabla 6. La Figura 6 muestra la GUI funcional conectada al sistema físico, mientras que la Figura 7 ilustra las gráficas de exactitud y pérdida generadas automáticamente tras el entrenamiento.

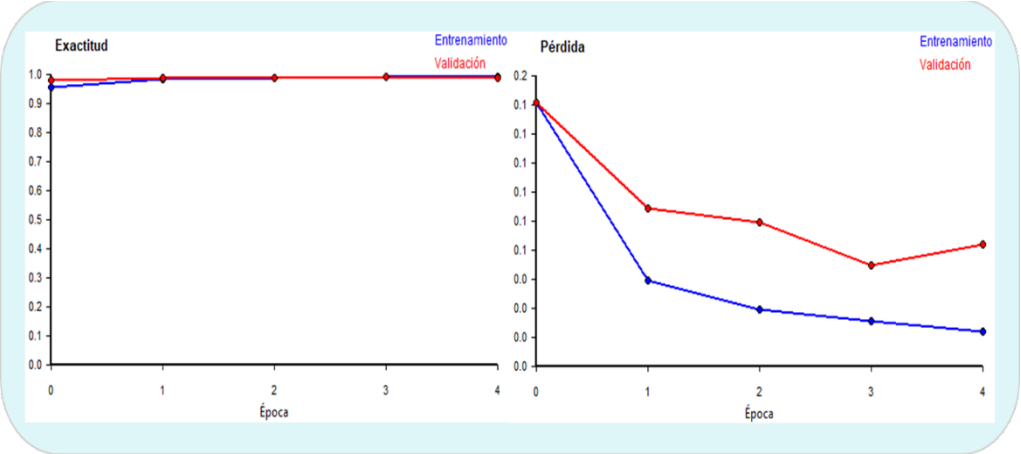

Figura 7. Resultados de Exactitud y Pérdida iniciales.

3.2. Segunda evaluación (optimización media)

Se repitió el proceso de entrenamiento y validación tras ajustar parámetros de tasa de aprendizaje, épocas y umbral T-Max-Avg. En esta segunda prueba, los resultados mostraron una ligera reducción en pérdida, con una precisión de entrenamiento de 99.02 % y prueba de 98.77 %. La precisión en pruebas reales superó los 99.5 % en promedio (ver Tabla 7).

Tabla 7. Resultados de la segunda evaluación.

Tipo de Medición	Precisión (%) / Pérdida	Promedio de Precisión (%)
Entrenamiento	99.02/0.0319	–
Prueba	98.77/0.0401	–
Predicción de Dígitos Real (1, 2, 3, 4)	99.65, 99.32, 99.31, 99.80	99.52

La Figura 8, muestra la vista de la GUI con los datos de configuración utilizados y los resultados obtenidos en la evaluación intermedia.

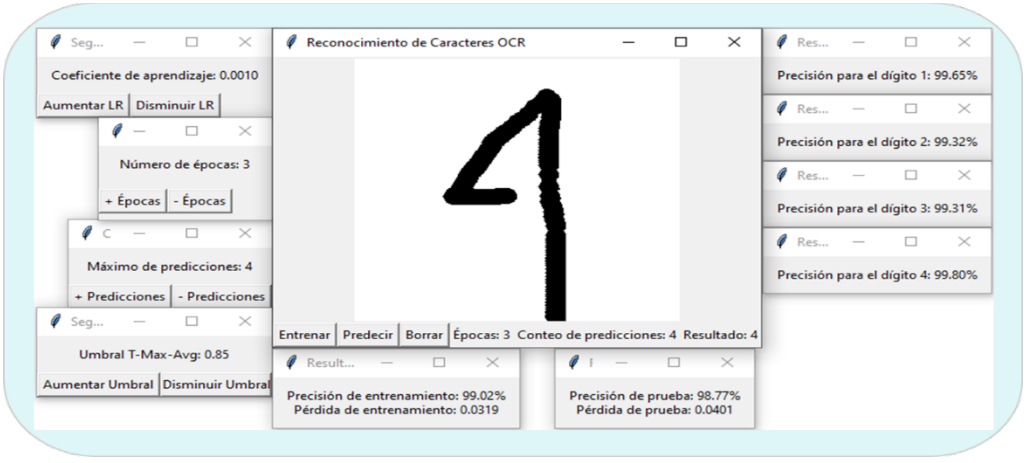

Figura 8. Segunda evaluación de funcionalidad de la interfaz gráfica.

Las gráficas relacionadas con el entrenamiento y prueba, en relación con la pérdida y exactitud, reflejan visualmente el comportamiento del modelo durante esta segunda iteración.

Las respuestas se muestran en la Figura 9.

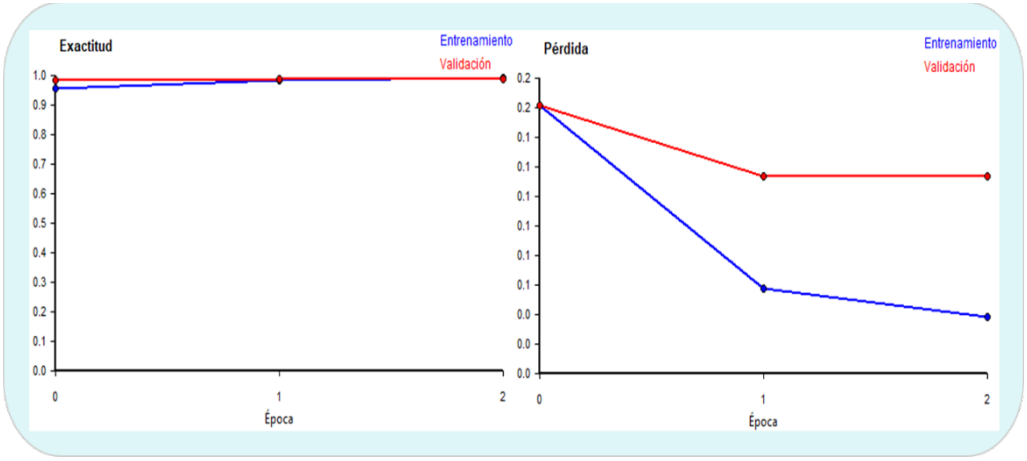

Figura 9. Resultados de Exactitud y Pérdida del segundo experimento.

3.3. Evaluación final

En la evaluación final se buscó validar la estabilidad del sistema, replicando condiciones de entrada más diversas (trazos más irregulares, diferentes usuarios).

Tabla 8. Resultados de la última evaluación.

Tipo de Medición	Precisión (%) / Pérdida	Promedio de Precisión (%)
Entrenamiento	99.08/0.0292	_
Prueba	98.72/0.0392	_
Predicción de Dígitos Real (1, 2, 3, 4)	99.21, 98.84, 99.01, 98.78	98.96

Los resultados se mantuvieron robustos, con una precisión de entrenamiento de 99.08 %, prueba de 98.72 % y una precisión media de 98.96 % para predicción manual (ver Tabla 8). La Figura 10 muestra la evolución funcional de la GUI tras esta evaluación.

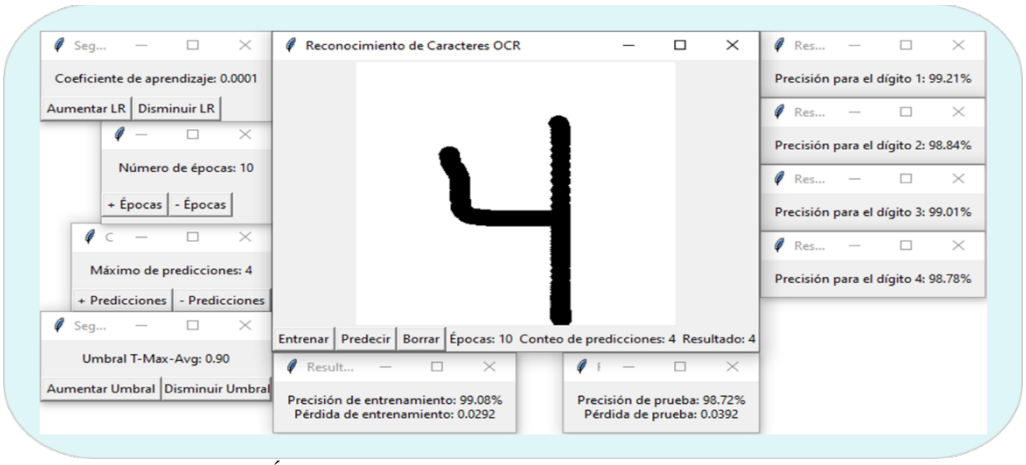

Figura 10. Última evaluación de funcionalidad de la interfaz gráfica.

Los ajustes realizados en los parámetros del Umbral, Número de épocas, pero principalmente en el Coeficiente de aprendizaje (valor muy cercano a cero), podrían llevar al algoritmo de clasificación a un estado de sobreajuste e inestabilidad.

Esta situación podría comprometer el reconocimiento correcto de los dígitos manuscritos, generando claves de acceso incorrectas.

En la Figura 11, se observan los gráficos generados con las repuestas de exactitud y pérdida, en relación con el último experimento.

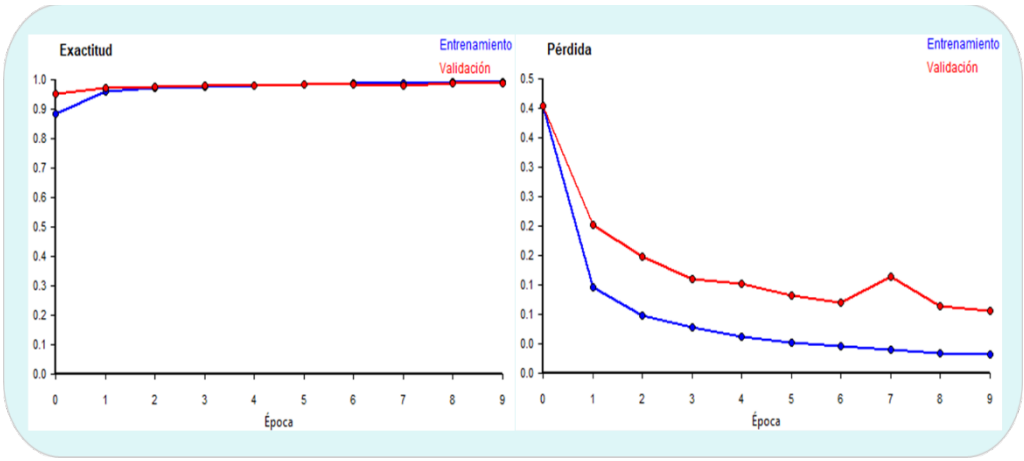

Figura 11. Resultados de Exactitud y Pérdida del experimento final.

Es importante comentar que las actuaciones, están relacionadas con el entrenamiento de los datos y su validación (pruebas).

3.4. Análisis comparativo de las métricas

En las tres evaluaciones, el modelo mostró una alta estabilidad y generalización, con variaciones mínimas en precisión y pérdida, tanto en conjunto de prueba como en predicciones manuales. Esto evidencia que la arquitectura CNN con T-Max-Avg, implementada en la GUI y combinada con entrenamiento dinámico por usuario, es eficaz y confiable para la generación de claves manuscritas en entornos reales.

4. Discusión

Los resultados obtenidos en las tres evaluaciones muestran una alta eficacia del sistema propuesto para el reconocimiento de dígitos manuscritos con fines de control de acceso. Las precisiones superiores al 98.7 % en conjunto de prueba y al 99.2 % en predicción de dígitos reales evidencian que el modelo CNN con T-Max-Avg puede generalizar adecuadamente a entradas no vistas. Además, el bajo nivel de pérdida durante el entrenamiento sugiere que no hay sobreajuste significativo.

Estos resultados son consistentes con estudios recientes que reportan mejoras en precisión mediante el uso de operadores híbridos de agrupamiento y técnicas de optimización evolutiva [8,9]. En particular, la integración de la red con una GUI dinámica y una plataforma de hardware accesible como Arduino permite su aplicación en contextos académicos, industriales o comunitarios.

Cabe destacar que el sistema responde correctamente a entradas físicas mediante Dispositivo Táctil, y que las gráficas de precisión y pérdida ofrecidas en la GUI permiten al usuario interpretar el rendimiento del modelo sin conocimientos técnicos avanzados. Este tipo de visualización y control directo favorece la adopción de inteligencia artificial en entornos donde la usabilidad es prioritaria.

Finalmente, se reconoce que el sistema puede ser mejorado mediante la incorporación de nuevos módulos, como autenticación biométrica dual o menús adaptativos. Futuros trabajos pueden enfocarse en la validación con una población mayor,

la inclusión de alfabetos no latinos, o la integración con servicios web para administración remota de accesos.

5. Conclusiones

La interfaz gráfica desarrollada, en conjunto con una red neuronal convolucional optimizada, demostró ser una herramienta efectiva para el reconocimiento de dígitos manuscritos y su aplicación en sistemas de control de acceso. La precisión obtenida en las pruebas tanto automáticas como reales superó el 98.9 %, evidenciando la robustez del modelo incluso en condiciones de entrada no ideal. La integración del sistema con una plataforma de hardware embebido como Arduino, y su operabilidad a través de una GUI intuitiva, ofrece una solución asequible y versátil para entornos educativos o institucionales. Además, la posibilidad de ajustar parámetros clave desde la interfaz otorga flexibilidad y personalización al sistema. Como línea futura, se propone ampliar la funcionalidad de la GUI con nuevas ventanas interactivas y extender el sistema para el reconocimiento de caracteres alfabéticos o símbolos especiales, así como su validación en poblaciones diversas.

Contribución: Conceptualización, J.J.C.M. y R.C.R.; metodología, J.J.C.M.; desarrollo de software, J.J.C.M.; validación, R.C.R. y B.M.B.; redacción del borrador, J.J.C.M.; revisión y edición, R.C.R., B.M.B. y J.J.C.M. Todos los autores leyeron y aprobaron la versión final del manuscrito. La autoría se limita a quienes contribuyeron sustancialmente al trabajo presentado.

Financiamiento: Esta investigación no recibió financiamiento externo.

Agradecimientos: Los autores agradecen al Instituto Tecnológico Superior de Huichapan por el respaldo institucional, así como a los estudiantes del Departamento de Mecatrónica que colaboraron en las pruebas experimentales y validación funcional del sistema.

Conflicto de interés: Los autores declaran no tener conflicto de intereses. Los financiadores no tuvieron ningún rol en el diseño del estudio; en la recopilación, análisis o interpretación de datos; en la redacción del manuscrito, o en la decisión de publicar los resultados.

Referencias

1. Peck, J.; Goossens, B.; Saeys, Y. An Introduction to Adversarially Robust Deep Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 2071-2090. https://doi.org/10.1109/TPAMI.2023.3331087
2. Wen, Y.; Ke, W.; Sheng, H. Improved Localization and Recognition of Handwritten Digits on MNIST Dataset with ConvGRU. *Appl. Sci.* **2025**, *15*, 1-16. https://doi.org/10.3390/app15010238
3. Xu, P.; Zhu, X.; Clifton, D. Multimodal Learning With Transformers: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 12113-12132. https://doi.org/10.1109/TPAMI.2023.3275156
4. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278-2324. https://doi.org/10.1109/5.726791
5. Tuba, I.; Tuba, U.; Veinović, M. Classification methods for handwritten digit recognition: A survey. *Mil. Tech. Gaz.* **2023**, 71, 113-135. https://doi.org/10.5937/vojtehg71-36914
6. Fateh, A.; Birgani, R.T.; Fateh, M.; Abolghasemi, V. Advancing Multilingual Handwritten Numeral Recognition with Attention-Driven Transfer Learning. *IEEE Access* **2024**, *12*, 41381-41395. https://doi.org/10.1109/ACCESS.2024.3378598
7. Lan, W.; Cheung, Y.; Jiang, J.; Hu, Z.; Li, M. Compact Neural Network via Stacking Hybrid Units. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, 46, 103-116. https://doi.org/10.1109/TPAMI.2023.3323496
8. Sarhadi, A.; Ravanshadnia, M.; Monirabbas, A.; Ghanbari, M. Using an improved U-Net++ with a T-Max-Avg-Pooling layer as a rapid approach for concrete crack detection. *Front. Built Environ.* **2024**, *10*, 1-13. https://doi.org/10.3389/fbuil.2024.1485774
9. Hussain, W.; Mushtaq, M.; Shahroz, M.; Akram, U.; Ghith, E.; Tlija, M.; Kim, T.; Ashraf, I. Ensemble genetic and CNN model-based image classification by enhancing hyperparameter tuning. *Sci. Rep.* **2025**, *15*, 1-24. https://doi.org/10.1038/s41598-024-76178-3
10. Ahmed, S.; Alam, M.; Hassan, M.; Rozbu, M.; Ishtiak, T.; Rafa, N.; Mofijur, S.; Ali, A.; Gandomi, A. Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artif. Intell. Rev.* **2023**, *56*, 13521-13617. https://doi.org/10.1007/s10462-023-10466-8
11. Cai, B.; Luo, J.; Feng, Z. A novel code generator for graphical user interfaces. *Sci. Rep.* **2023**, *13*, 1-16. https://doi.org/10.1038/s41598-023-46500-6

12. Jeon, S.; Ko, B.; Son, S. ROMI: A Real-Time Optical Digit Recognition Embedded System for Monitoring Patients in Intensive Care Units. *Sensors* **2023**, *23,* 1-24. https://doi.org/10.3390/s23020638
13. Jain, P.; Kumar, V.; Samuel, J.; Singh, S.; Mannepalli, A.; Anderson, R. Artificially Intelligent Readers: An Adaptive Framework for Original Handwritten Numerical Digits Recognition with OCR Methods. *Information* **2023**, *14,* 1-19. https://doi.org/10.3390/info14060305
14. Chaves, A.; Mendonça, F.; Mostafa, S.; Morgado-Dias, F. Graphical User Interface for the Development of Probabilistic Convolutional Neural Networks. *Signals* **2023**, *4,* 297-314. https://doi.org/10.3390/signals4020016
15. Salkanovic, A.; Sušanj, D.; Batistić, L.; Ljubic, S. Beyond Signatures: Leveraging Sensor Fusion for Contextual Handwriting Recognition. *Sensors* **2025**, *25*, 1-29. https://doi.org/10.3390/s25072290
16. Tolosana, R.; Vera-Rodriguez, R.; Fierrez, J. BioTouchPass: Handwritten Passwords for Touchscreen Biometrics. *IEEE Trans. Mobile Comput.* **2020**, *19*, 1532-1543. https://doi.org/10.1109/TMC.2019.2911506
17. Lin, C.; Zhouhe, S.; Ahmad, A.; Fan, Xinxin.; Wang, Y.; Wang, L.; Fan, Xin; Wu, G. AirWrite: An Aerial Handwriting Trajectory Tracking and Recognition System With mmWave. *IEEE Trans. Mobile Comput.* **2024**, *23*, 13325-13341. https://doi.org/10.1109/TMC.2024.3425709
18. Rahman, M.; Siddiqui, A.; Erad, J.; Rahman, T.; Chowdhury, N.; Bhuyan, M. A Multi-Functional Arduino-Based Autonomous Vehicle for Landmine and Gas Detection. In Proceedings of the 2025 International Conference on Electrical, Computer and Communication Engineering, Chittagong, Bangladesh, (13-15 February 2025). https://doi.org/10.1109/ECCE64574.2025.11013130
19. Arduino community. Available online: https://www.arduino.cc/ (accessed on 01 August 2025).
20. Köhli, M.; Weimar, J.; Schmidt, S.; Schmidt, F.; Lambertz, A.; Weber, L.; Kaminski, J.; Schmidt, U. Arduino-Based Readout Electronics for Nuclear and Particle Physics. *Sensors* **2024**, *24,* 1-18. https://doi.org/10.3390/s24092935
21. Inik, Ö. SwarmCNN: An efficient method for CNN hyperparameter optimization using PSO and ABC metaheuristic algorithms. *J. Supercomput.* **2025**, *81,* 1-42. https://doi.org/10.1007/s11227-025-07347-y
22. Zhao, X.; Wang, L.; Zhang, Y.; Han, X.; Deveci, M.; Parmar, M. A review of convolutional neural networks in computer vision. *Artif. Intell. Rev.* **2024**, *57,* 1-43. https://doi.org/10.1007/s10462-024-10721-6
23. Zhao, L.; Zhang, Z. An improved pooling method for convolutional neural networks. *Sci. Rep.* **2024**, *14,* 1-22. https://doi.org/10.1038/s41598-024-51258-6
24. El-Khamy, S.; El-Bana, S.; Al-Kabbany, A.; Elragal, H. Toward better semantic segmentation by retaining spectral information using matched wavelet pooling. *Neural Comput. Appl.* **2025**, *37*, 7049-7066. https://doi.org/10.1007/s00521-025-11008-9
25. Alemayoh, T.; Shintani, M.; Lee, J.; Okamoto, S. Deep-Learning-Based Character Recognition from Handwriting Motion Data Captured Using IMU and Force Sensors. *Sensors* **2022,** *22,* 1-15. https://doi.org/10.3390/s22207840
26. Menke, J.; Nahal, Y.; Bjerrum, E.; Kabeshov, M.; Kaski, S.; Engkvist, O. Metis: a Python-based user interface to collect expert feedback for generative chemistry Models. *J. Cheminform.* **2024**, *16*, 1-9. https://doi.org/10.1186/s13321-024-00892-3
27. Raza, S.; Farooq, M.; Farooq, U.; Karamti, H.; Khurshaid, T.; Ashraf, I. A Convolutional Neural Network Based Optical Character Recognition for Purely Handwritten Characters and Digits. *Comput. Mater. Contin.* **2025**, *84,* 3149-3173. https://doi.org/10.32604/cmc.2025.063255
28. Krichen, M. Convolutional Neural Networks: A Survey. *Computers* **2023**, *12,* 1-41. https://doi.org/10.3390/computers12080151