

De 7.2 M a 532 k parámetros: creación y optimización de una CNN para billetes de Bangladesh

Sergio Luis Agreda Lemuz

Universidad Mayor de San Andrés, Bolivia

sergioagreda21@outlook.com

Abstract—Este proyecto presenta el desarrollo de una Red Neuronal Convolucional (CNN) para la clasificación de billetes de Bangladesh de diferentes denominaciones. El modelo propuesto está diseñado para aprender y reconocer patrones distintivos en los billetes, permitiendo su identificación a partir de una sola captura de imagen. Las posibles aplicaciones del reconocimiento de billetes van desde el sector financiero, siendo esencial como un medio de seguridad financiera para el reconocimiento de billetes falsos, velocidad y seguridad de clasificación de billetes para sistemas de transacciones electrónicas, hasta incluir la clasificación de billetes para la mejora de calidad de vida de gente ciega [1]–[3]. La integración de un sistema de reconocimiento basado en CNN permitiría detectar y clasificar automáticamente los billetes, optimizando la eficiencia operativa y reduciendo la intervención humana.

Index Terms—Convolutional Neural Networks, Deep Learning, Clasificación de Imágenes, Banknotes, Reconocimiento Automático.

I. INTRODUCCIÓN

AS redes neuronales profundas (*Deep Neural Networks, DNN*) son redes neuronales artificiales caracterizadas por su profundidad, es decir, múltiples capas ocultas entre las fases de entrada y salida. Esta complejidad estructural les permite modelar relaciones altamente no lineales y abstracciones jerárquicas en los datos, lo que es esencial para tareas avanzadas de predicción y reconocimiento. Estas propiedades han convertido a las DNN en una columna vertebral del aprendizaje profundo y de la inteligencia artificial moderna [4].

Parte de las redes neuronales profundas, y ya entrando a una mayor complejidad, se encuentran las redes neuronales convolucionales (*Convolutional Neural Networks, CNN*). Son una categoría especializada dentro de las DNN, diseñada para procesar datos con estructura espacial, como imágenes. Emplean filtros convolucionales que se deslizan sobre la imagen para extraer características locales, como bordes y texturas, y lo hacen de forma jerárquica, aportando una alta capacidad para capturar patrones espaciales sin necesidad de extracción manual de características [4].

Las redes convolucionales (CNN) destacan por su aptitud para generalizar desde grandes cantidades de datos etiquetados, aprendiendo representaciones que se vuelven progresivamente más abstractas conforme avanzan por las capas de red. Esta

capacidad hace posible alcanzar notables niveles de precisión en reconocimiento de objetos, clasificación y segmentación de imágenes, incrementando la robustez frente a variaciones de entrada [5].

El avance de las DNN y CNN ha sido impulsado por mejoras en hardware (como GPUs), disponibilidad de grandes bases de datos etiquetados (como ImageNet) y mejoras arquitectónicas como las redes residuales (ResNet), que facilitan el entrenamiento de redes más profundas y precisas [6].

La importancia fundamental de estas redes radica en su habilidad para aprender automáticamente representaciones complejas directamente desde los datos sin intervención humana intensiva, lo cual resuelve el problema de extracción de características manual. Esto ha permitido automatizar tareas anteriormente limitadas a intervención experta o programación artesanal, generando un profundo impacto en múltiples industrias [7].

En este proyecto se desarrolla una CNN personalizada para el reconocimiento de billetes de Bangladesh en nueve denominaciones diferentes. Esta solución va en línea con varias investigaciones ya realizadas [1], [2], [8] y responde a las necesidades de negocios que operan principalmente con moneda física, en contraposición a los métodos de pago digitales como códigos QR, tarjetas de crédito/débito o aplicaciones móviles. La implementación de este modelo podría automatizar y agilizar procesos en sectores donde las transacciones en efectivo siguen siendo predominantes.

II. ESTADO DEL ARTE

El reconocimiento automatizado de divisas ha pasado de sistemas basados en reglas y sensores, que dependían de características diseñadas manualmente (como señales UV/IR, bordes o descriptores de textura), a sistemas de visión por computadora impulsados por datos. Los algoritmos tradicionales solían tener dificultades para generalizar en condiciones reales, enfrentándose a problemas como el desgaste de los billetes, manchas o cambios en la iluminación. Además, requerían una cuidadosa ingeniería de características y ajustes específicos para cada dispositivo. Las Redes Neuronales Convolucionales (CNN) han sustituido estos métodos al aprender directamente características jerárquicas a partir de imágenes, lo que les permite manejar variaciones dentro de una misma clase, aprovechar el aprendizaje por transferencia desde grandes colecciones de imágenes y, cada vez más, implementarse en

hardware embebido de forma ligera. Por ello, la investigación actual se enfoca en la precisión, la eficiencia y la capacidad de desplegarse en tiempo real para distintas denominaciones, condiciones de billetes y conjuntos de divisas de diferentes regiones.

Los estudios recientes basados en CNN muestran dos tendencias complementarias: por un lado, el desarrollo de arquitecturas ligeras especializadas para la clasificación de múltiples divisas; por otro, el uso de aprendizaje por transferencia para la detección de billetes falsos. En el primer caso [8], presentan CA-DSC-RepVGG, una versión mejorada de RepVGG-A0 que incorpora atención por coordenadas y convoluciones separables en profundidad, con el objetivo de aumentar la precisión reduciendo al mismo tiempo el número de parámetros. Entrenado sobre el conjunto de datos Multi-Currency Banknote Dataset (AUD/EUR/USD; 91,200 imágenes), el modelo se beneficia del preentrenamiento en ImageNet y de técnicas de optimización estándar, obteniendo un alto rendimiento tanto en validación como en pruebas. Al implementarse en hardware embebido (Rockchip rk3568), mantiene una velocidad aproximada de 101,7 cuadros por segundo con 9,83 ms por imagen, mejorando la precisión en $\sim 1,05\%$ respecto a su modelo base y reduciendo los parámetros en un $\sim 78,4\%$ [8]. En cambio, Mondal [1] aborda la detección de falsificaciones como un problema de clasificación binaria, evaluando varias CNN preentrenadas (VGG-19, InceptionResNet-V2, Inception-V3, ResNet-50) sobre un conjunto de datos de Bangladesh con 6,113 imágenes (500/1000 Tk), usando una división 80/20, redimensionamiento a 300×300 y aumentos de datos estándar. Entre las arquitecturas probadas, VGG-19 obtiene la mayor precisión en validación (95,47%) [1].

En el plano metodológico, cada trabajo responde a diferentes limitaciones y objetivos. Unos [8] se enfocan en el reconocimiento de denominaciones de múltiples divisas bajo restricciones de implementación industrial, proponiendo un diseño con atención optimizada, convoluciones separables y reparametrización estructural para equilibrar precisión y eficiencia de inferencia en hardware limitado. Su protocolo de entrenamiento sobre MCBD (RepVGG-A0 inicializado en ImageNet, optimizador Adam, entropía cruzada, lote de 16, entradas de 224×224 , *early stopping* y programación de calentamiento y decaimiento) logra precisiones cercanas al máximo posible en AUD/EUR/USD, con una ligera caída en el billete de EUR 200. Por su parte, Mondal [1] demuestra la viabilidad del aprendizaje por transferencia para la detección de billetes falsos en entornos con pocos datos, ajustando modelos base entrenados en ImageNet. Reportan tanto resultados en validación (VGG-19 con 95,47%) como una pequeña prueba independiente que ilustra las relaciones entre precisión y *recall* (88% de precisión en 50 imágenes), identificando fallos bajo variaciones de iluminación, enfoque y en billetes deteriorados. Además, reconocen que su conjunto de datos se construyó combinando billetes genuinos capturados con teléfonos inteligentes y falsificaciones elaboradas con Photoshop e impresión, lo que supone un compromiso práctico con implicaciones sobre el realismo [1].

En comparación, los métodos anteriores basados en ingeniería de características todavía sirven como referencia para

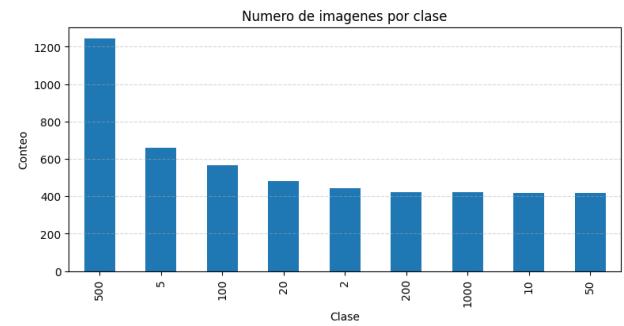


FIGURA 1. Distribución de clases (billetes).

la evaluación del estado de los billetes. Tong [2] propone un sistema rápido basado en imágenes que extrae estadísticas de textura a partir de la Matriz de Co-ocurrencia de Gradiente de Nivel de Gris (GLGCM) y clasifica los billetes en cinco niveles de desgaste utilizando un DAG-SVM multiclas. En un conjunto de datos RMB-100, alcanzan más del 99% de precisión en todos los niveles, con un rendimiento adecuado para líneas de clasificación industrial (8–9 segundos por cada 1,000 billetes). También se señalan las limitaciones de los enfoques previos: las redes neuronales de entonces se consideraban demasiado costosas computacionalmente para entornos de alto rendimiento, mientras que los métodos puramente estadísticos podían perder precisión. Esto refuerza la motivación para las CNN modernas, que logran combinar velocidad y exactitud [2], [8].

En conjunto, estos estudios trazan una evolución clara: desde características diseñadas manualmente para sistemas en tiempo real, pasando por CNN con aprendizaje por transferencia para detectar falsificaciones con pocos datos, hasta arquitecturas ligeras con mecanismos de atención pensadas para reconocer múltiples divisas a velocidades industriales en hardware embebido. Entre los retos comunes identificados se encuentran la similitud entre clases y la confusión de denominaciones (con caídas de precisión en ciertos billetes), el realismo y cobertura de los datos para ejemplares falsos (fabricados versus incautados) y las limitaciones de implementación en dispositivos con recursos restringidos. Los resultados sugieren que el uso de mecanismos de atención, convoluciones separables y un preentrenamiento y regularización bien diseñados pueden ayudar a superar estos desafíos, manteniendo al mismo tiempo la operación en tiempo real.

III. METODOLOGÍA

El modelo se desarrolló utilizando *Jupyter Notebook* y la librería *TensorFlow*. El entorno de trabajo fue Ubuntu, ejecutado mediante WSL2 en Windows para habilitar la aceleración por GPU. Las especificaciones de hardware incluyeron 24 GB de memoria RAM, una tarjeta gráfica *NVIDIA GTX 1650* con 4 GB de VRAM y un procesador *Intel Core i7* de 9.^a generación.

El *dataset* constó de 5,000 imágenes (1,71 GB) organizadas en nueve carpetas, cada una correspondiente a una denominación específica: 2, 5, 10, 20, 50, 100, 200, 500 y 1000 Taka (moneda oficial de Bangladesh). La distribución

de imágenes por clase fue la siguiente: 1,243 imágenes para billetes de 500 Taka, 660 para billetes de 5 Taka, 565 para billetes de 100 Taka, 480 para billetes de 20 Taka, 445 para billetes de 2 Taka, 423 para billetes de 200 Taka, 422 para billetes de 1000 Taka, 419 para billetes de 10 Taka y 416 para billetes de 50 Taka. Esta distribución evidencia un marcado desbalance de clases, con una notable predominancia de la clase correspondiente a billetes de 500 Taka frente a las demás. Asimismo, se observa un leve desbalance en las clases de billetes de 5 y 100 Taka respecto a las restantes. Para mitigar este problema se aplicó la técnica de `class_weights` durante el entrenamiento, con el fin de ajustar la importancia relativa de cada clase, otorgando mayor ponderación a aquellas con menor cantidad de imágenes. En la Figura 1 se muestra la distribución de imágenes por clase. Los datos se dividieron en dos subconjuntos principales: `trainvalSplit` (90% del dataset) y `testSplit` (10%). El subconjunto `trainvalSplit` se subdividió a su vez en entrenamiento (70%) y validación (20%). En la Tabla I se presenta una descripción resumida de las clases del conjunto de datos.

Debido a la gran variabilidad en las resoluciones de las imágenes, que en más del 50% de los casos superaban los 1800×1200 píxeles, todas fueron redimensionadas a 299×299 píxeles, preservando la relación de aspecto original y reduciendo su tamaño hasta que se ajustara a la dimensión establecida. En aquellas imágenes cuya relación de aspecto difería de 299×299 píxeles se añadieron bordes negros cuando fue necesario.

Se aplicaron técnicas de aumento de datos exclusivamente al conjunto de entrenamiento para mejorar la capacidad de generalización del modelo. Se utilizaron transformaciones aleatorias como `RandomZoom` (acercamiento o alejamiento leve) y `RandomContrast` (ajuste de contraste).

La arquitectura de la CNN siguió un diseño secuencial para la clasificación de las nueve clases. Consistió en capas `Conv2D` con *kernels* de 3×3 y activación `ReLU`, intercaladas con capas `MaxPooling2D` de 2×2 para la reducción espacial. Posteriormente, se empleó una capa `GlobalAveragePooling2D` para convertir los mapas de características en un vector unidimensional, seguida de una capa `Dropout(0.4)` y, finalmente, una red densa de tres capas completamente conectadas con 256, 128 y 64 neuronas, respectivamente, todas con activación `ReLU`. La capa de salida estuvo compuesta por nueve neuronas con activación `Softmax` para producir la distribución de probabilidades de cada clase.

IV. RESULTADOS

El desarrollo de la Red Neuronal Convolutacional (CNN) para clasificar billetes de Bangladés se llevó a cabo mediante cinco fases (cuatro experimentales), ajustando la arquitectura y las estrategias de entrenamiento para equilibrar precisión y eficiencia computacional.

V. RESULTADOS

El desarrollo de la Red Neuronal Convolutacional (CNN) para clasificar billetes de Bangladés se llevó a cabo mediante



FIGURA 2. Imágenes de billetes del conjunto de datos.

cinco fases (cuatro experimentales), ajustando la arquitectura y las estrategias de entrenamiento para equilibrar precisión y eficiencia computacional.

Primera ejecución: Que posteriormente sirvió como línea de referencia. Constaba de una arquitectura con varias capas `Conv2D` y `MaxPooling2D` intercaladas, regularización mínima y una capa `Flatten()` antes de las capas densas, generando un modelo con 7,2 millones de parámetros. Tras 25 épocas, alcanzó 97,38% de *accuracy* en entrenamiento y 97,81% en validación, con métricas sólidas (precisión, *recall* y *F1*) a través de las nueve clases. En el conjunto de prueba (*test*) obtuvo 97,8% de *accuracy* con pérdidas bajas (*loss* en entrenamiento = 0,0937, *loss* en validación = 0,0984). A pesar de haber obtenido muy buenos resultados, tenía una desventaja: su alto costo computacional, lo que motivó la búsqueda de un modelo más ligero.

Segunda ejecución: Sustitución de capa `Flatten()` por `GlobalAveragePooling2D()` para reducir parámetros e incorporación de *callbacks* como `EarlyStopping` y `ModelCheckpoint`. La introducción de `GlobalAveragePooling2D()` llevó a una reducción de parámetros, los cuales descendieron abruptamente a $\sim 186\,000$. Sin embargo, el rendimiento también sufrió un descenso: 92,99% en validación y 92,3% en prueba (*test*). La limitación radicó en la capacidad insuficiente de las capas densas posteriores, que no tenían suficiente capacidad de representación.

Tercera ejecución: Ampliación de las capas densas (se incrementó el doble de neuronas por capa) y adición de `ReduceLROnPlateau` para ajustar la tasa de aprendizaje en el entrenamiento. Se intentó cambiar el *padding* de *valid* a *same* en las capas `Conv2D`, aunque por error se aplicó en `MaxPooling2D` en lugar de `Conv2D`. Aun así, el rendimiento mejoró: 95,62% en validación y 94,8% en prueba (*test*), con $\sim 225\,000$ parámetros.

Cuarta ejecución: Se añadieron más capas de aumento de datos y `Dropout(0.3)` tras `GlobalAveragePooling2D`. Sin embargo, el rendimiento cayó a 92,77% de *accuracy* en validación y 92,3% en prueba. La disminución en el rendimiento del modelo se atribuyó principalmente a la estrategia de adición de más capas de aumento de datos, excesivamente agresiva, aplicada al conjunto de entrenamiento. Se determinó que esta técnica, en lugar de mejorar la generalización del modelo, perjudicó su desempeño. Aunque se consideró la capa de

TABLA I
DESCRIPCIÓN DE CLASES

Directorio	Folder/Clase	Número de imágenes	Imagen
	2	445	
Bangladeshi_Paper_Currency_Raw	5	660	
	10	419	
	20	480	
	50	418	
	100	566	
	200	423	
	500	1243	
	1000	421	

dropout como una posible causa, la hipótesis principal apuntó al aumento de datos como el factor determinante de la caída del rendimiento.

Quinta ejecución (seleccionada): Dada la hipótesis de la ejecución cuatro, se retiraron las capas extra de aumento de datos, manteniendo *dropout*. El *accuracy* subió a 96,93% en validación y 97,4% en prueba (*test*), con ~532 000 parámetros, prácticamente igualando la primera ejecución en resultados,

pero con una huella 15× menor. Aunque la pérdida fue más alta (pérdida en validación actual = 0,3663 vs. 0,0984 en primera ejecución), la matriz de confusión y métricas por clase fueron comparables o mejores para varias clases, sugiriendo menor calibración de confianza más que errores de clasificación.

Sexta ejecución: Tras la ejecución número cinco, en la que se logró reducir los parámetros a una quinceava parte respecto a la primera ejecución manteniendo métricas equivalentes, y una

TABLA II
RESUMEN COMPARATIVO DE EJECUCIONES

Run	Params	Epochs	Val Acc	Val Loss	Test Acc	Test Loss	Macro-F1 (test)	Weighted-F1 (test)
1	7.2M	25	0.9781	0.0984	0.978	0.101	0.98	0.98
2	186k	25	0.9299	0.2214	0.923	0.223	0.92	0.92
3	225k	44	0.9562	0.1313	0.948	0.169	0.95	0.95
4	533k	27	0.9277	0.2236	0.923	0.218	0.91	0.92
5*	533k	55	0.9693	0.3663†	0.974	0.365†	0.97	0.97

Notas: * Modelo seleccionado. † Pérdida más alta que la línea base; ver discusión sobre calibración.

vez identificados los hiperparámetros y la arquitectura óptima de la red convolucional, se aplicó validación cruzada de 5 pliegues (*5-fold Cross-Validation*). El objetivo de esta etapa fue verificar que el rendimiento observado en la quinta ejecución no fuera producto de un sesgo en la partición inicial de los datos, sino que representara un comportamiento consistente y generalizable a diferentes subconjuntos del conjunto de datos. Este procedimiento permitió evaluar la estabilidad de las métricas, reducir la varianza de la estimación de rendimiento y obtener una medida más robusta de la capacidad de generalización del modelo. En esta fase también se implementó la técnica de *class weights* mencionada anteriormente para abordar el desbalance de clases. Los resultados fueron estables: si bien la introducción de esta estrategia no produjo mejoras significativas en las métricas, tampoco las degradó. Se obtuvo un rango promedio de *accuracy* de 96,7–97,4% tanto en validación como en entrenamiento, y un rango promedio de *loss* de 0,1092–0,217, este último mostrando una ligera mejora respecto a la quinta ejecución.

En conclusión, la progresión de la primera a la sexta ejecución demostró un cambio exitoso en la compensación entre el tamaño del modelo y la exactitud del mismo. Mientras que la primera ejecución mostró valores de exactitud ligeramente más altos en validación, entrenamiento y prueba (*test*), y con valores de pérdida más bajos, la quinta ejecución igualó o superó en exactitud y otras métricas a nivel de clases, siendo un modelo con una cantidad 15 veces menor de parámetros. Con la sexta ejecución se confirmó que los resultados de la quinta ejecución no fueron fortuitos, sino consistentes y generalizables, validando la capacidad del modelo para mantener la misma precisión en la clasificación de este tipo de imágenes a pesar de su menor complejidad. Esto resultó en un modelo mucho más eficiente computacionalmente. Dadas estas mejoras en la eficiencia del modelo, y con una pérdida mínima de *accuracy*, se seleccionó la quinta ejecución como la arquitectura preferida para el modelo CNN.

En la Tabla II se puede ver una comparación de resultados y métricas de las seis ejecuciones que se realizaron en este estudio. Asimismo, se pueden visualizar los gráficos de *loss* en entrenamiento y validación, así como los de *accuracy* en las Figuras 4 y 5. Por último, se puede ver en la Figura 3 la matriz de confusión que obtuvo la quinta ejecución (el modelo seleccionado).

VI. DISCUSIÓN

El proceso de desarrollo iterativo del modelo convolucional para la clasificación de la moneda bangladesí puso en relieve la relación entre la complejidad del modelo, el rendimiento y la eficiencia computacional. La diversidad del conjunto de datos en cuanto a resolución y calidad hizo necesaria una estrategia de preprocessamiento que normalizara los tamaños de entrada (299×299 px) y preservara la relación de aspecto, asegurando que las características finas de los billetes, así como patrones y detalles específicos de cada denominación, siguieran siendo distinguibles para el modelo.

Los resultados obtenidos en las diferentes ejecuciones evidenciaron que es posible mantener un nivel de *accuracy*

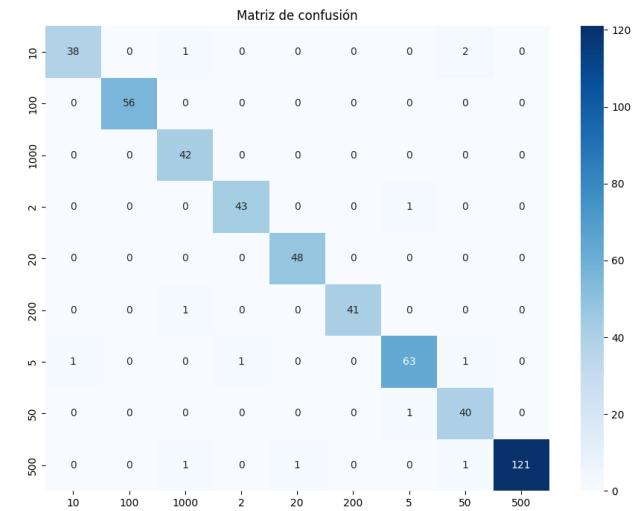


FIGURA 3. Matriz de confusión.

muy cercano al de un modelo de gran tamaño, pero reduciendo significativamente su número de parámetros. En particular, la quinta ejecución logró igualar o superar las métricas de varias clases respecto a la primera ejecución, reduciendo la complejidad del modelo en un factor de $15\times$, lo que representa una mejora sustancial en términos de eficiencia computacional y viabilidad de despliegue en entornos con recursos limitados.

El análisis de la sexta ejecución, con la aplicación de validación cruzada de 5 pliegues, permitió confirmar que los resultados obtenidos no fueron producto de un sesgo en la partición inicial de los datos, sino que se mantuvieron consistentes y generalizables a través de múltiples subconjuntos del conjunto de datos. Esta estabilidad en las métricas refuerza la solidez del modelo y su capacidad de generalización.

En cuanto al manejo del desbalance de clases, la incorporación de *class weights* no produjo mejoras significativas en la exactitud global, pero permitió mantener el rendimiento sin degradaciones, lo que sugiere que la arquitectura seleccionada es intrínsecamente robusta frente a distribuciones desbalanceadas.

Por último, la experimentación reveló que un aumento excesivamente agresivo de datos, como el implementado en la cuarta ejecución, puede perjudicar el rendimiento en lugar de mejorarlo, destacando la importancia de ajustar cuidadosamente el grado y tipo de transformaciones utilizadas para optimizar la capacidad de generalización del modelo.

Los primeros experimentos, ejecución uno, demostraron que una arquitectura más profunda y con un alto número de parámetros podía lograr un rendimiento excelente, con clasificaciones casi perfectas en la mayoría de las clases. Sin embargo, los 7,2 millones de parámetros implicaban un costo computacional alto, lo que limitaba su practicidad en entornos con recursos de hardware restringidos o de despliegue limitado. Las ejecuciones posteriores exploraron la reducción del tamaño del modelo sin sacrificar el *accuracy*.

Reemplazar `Flatten()` con `GlobalAveragePooling2D()` (segunda ejecución) redujo drásticamente los parámetros. Sin embargo, inicialmente

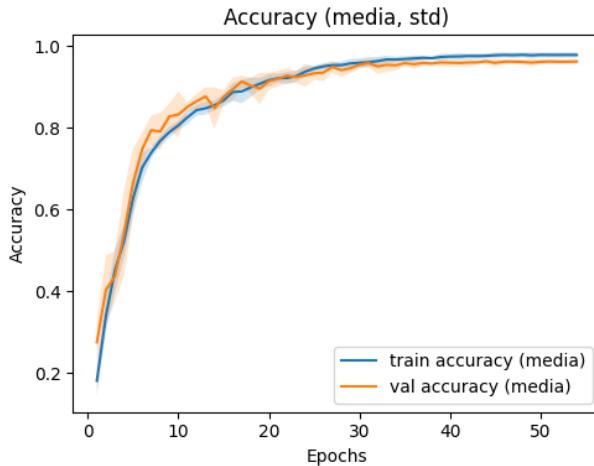


FIGURA 4. Grafico del rendimiento de accuracy durante el entrenamiento.

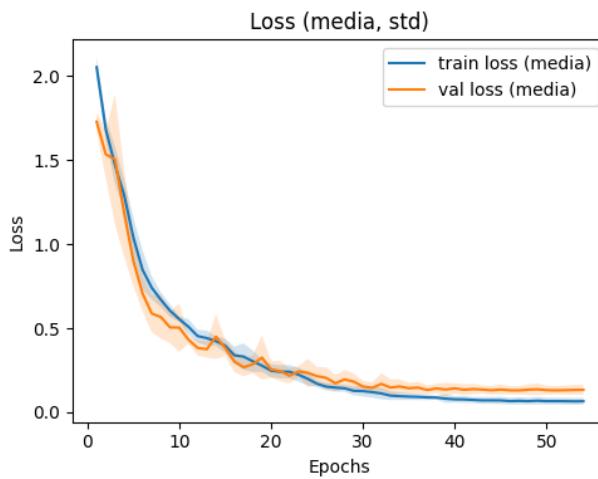


FIGURA 5. Grafico del cambio de loss durante el entrenamiento.

provocó una caída considerable del rendimiento debido a la insuficiente capacidad de las capas densas posteriores. El aumento de estas capas en la ejecución tres recuperó gran parte del rendimiento perdido, incluso a pesar del error de aplicación del parámetro *padding* en capas erróneas, lo que sugiere que la profundidad de la arquitectura y la capacidad de las capas densas fueron factores críticos para mantener la precisión.

La cuarta ejecución reveló sensibilidad a decisiones de *aumento de datos* y regularización: más no siempre es mejor; Si bien el *aumento de datos* es una técnica generalmente beneficiosa para enriquecer la diversidad del conjunto de entrenamiento, su aplicación excesivamente agresiva puede ser contraproducente. Una variabilidad tan extrema puede distorsionar las características esenciales de las imágenes, impidiendo que el modelo aprenda los patrones fundamentales necesarios para una clasificación correcta y, en consecuencia, perjudicando su rendimiento en lugar de mejorarlo. La (quinta ejecución) mantuvo *dropout* y logró el mejor equilibrio observado: precisión de prueba del 97,4% con una huella de parámetros muy reducida. La pérdida más alta sugiere menor

calibración de probabilidades, un aspecto mejorable mediante técnicas de calibración en trabajos futuros.

En conjunto, los resultados indican que una reducción deliberada de parámetros, con ajustes específicos en capas densas y una gestión prudente de las capas de *aumento de datos*, puede producir modelos CNN eficientes y ligeros, sin pérdidas significativas de precisión.

VII. CONCLUSIÓN

Este estudio demuestra que un modelo CNN para la clasificación multiclase de billetes puede optimizarse para alcanzar un rendimiento cercano al *state-of-the-art* con una fracción de la huella computacional inicial. Partiendo de 7,2 millones de parámetros, las iteraciones redujeron la complejidad y recuperaron el rendimiento mediante ajustes arquitectónicos, *hyperparameter tuning* y experimentación informada.

El modelo seleccionado (Ejecución 5) alcanzó 97,4% de *accuracy* en el conjunto de prueba con ~532 000 parámetros, lo que representa una ganancia sustancial en eficiencia sin comprometer la capacidad predictiva. Aunque la pérdida fue más alta que en la línea base, la matriz de confusión y las métricas por clase confirman una clasificación sólida en las nueve denominaciones.

La validación cruzada de 5 pliegues aplicada en la sexta ejecución permitió confirmar que estos resultados no fueron producto de un sesgo en la partición inicial, sino que se mantuvieron consistentes y generalizables a través de diferentes subconjuntos del conjunto de datos. Asimismo, la incorporación de *class weights* para abordar el desbalance de clases no mejoró las métricas globales, pero demostró que el modelo conserva su rendimiento incluso en escenarios de distribución desigual.

En conjunto, estos hallazgos evidencian que es posible diseñar un modelo de alta precisión y bajo costo computacional, apto para su implementación en entornos con recursos limitados y potencialmente aplicable a sistemas de reconocimiento de billetes en tiempo real.

REFERENCES

- [1] T. Mondal, P. Chakraborty, and S. I. Meem, “Bangladeshi CF guard: Unveiling bangladeshi counterfeit currency through transfer learning,” in *Innovative Computing and Communications*, A. E. Hassanien *et al.*, Eds. Singapore: Springer Nature Singapore, 2024, pp. 1–15.
- [2] C. Tong, Y. Lian, J. Qi, Z. Xie, A. Zhang, J. Feng, and F. Zhu, “A novel classification algorithm for new and used banknotes,” *Mobile Networks and Applications*, 2016.
- [3] M. N. I. Nuhash and S. Akter, “Banglataka: A dataset for classification of bangladeshi banknotes,” *Data in Brief*, vol. 61, p. 111853, Jul. 2025.
- [4] M. M. Taye, “Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions,” *Computation*, vol. 11, no. 3, p. 52, Mar. 2023.
- [5] A. Khan, A. Sohail, U. Zahoor, and A. S. Qureshi, “A review of convolutional neural networks in computer vision,” *Egyptian Informatics Journal*, vol. 22, no. 3, pp. 213–230, Nov. 2021.
- [6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
- [7] E. Jawad, “The deep neural network – a review,” *IJRDO - Journal of Mathematics*, vol. 9, no. 9, pp. 1–5, Sep. 2023.
- [8] X. Yang, Z. Zhao, K. Yuan, C. Xiao, and Y. Luo, “Mobile recognition system for multinational currencies based on CA–DSC–RepVGG algorithm,” *The Journal of Supercomputing*, vol. 81, no. 2, p. 433, Jan. 2025, art. no. 433.