

Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Electrónica

Identificación de billetes falsificados peruanos utilizando arquitecturas híbridas de aprendizaje profundo con CNN y RNN aplicadas a visión por computadora

Tesis presentada por:

Cardenas Mayma, Ray Kevin

ORCID: 0009-0008-0112-452X

para optar el Título Profesional de Ingeniero Electrónico

Asesor (a):

Mg. Urrutia Espinoza, Mario William

ORCID: 0000-0002-8520-2909

Arequipa - Perú

2026

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA ELECTRONICA

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 01 de Septiembre del 2026

Dictamen: 017580-C-EPIE-2026

Visto el borrador del expediente 017580, presentado por:

2016701981 - CARDENAS MAYMA RAY KEVIN

Titulado:

IDENTIFICACIÓN DE BILLETES FALSIFICADOS PERUANOS UTILIZANDO ARQUITECTURAS HÍBRIDAS DE APRENDIZAJE PROFUNDO CON CNN Y RNN APLICADAS A VISIÓN POR COMPUTADORA

Nuestro dictamen es:

APROBADO

Titulo Profesional/Titulo de Segunda Especialidad/Grado Académico a optar:

INGENIERO ELECTRONICO

29715414 - COAGUILA GOMEZ RONALD PERCING
DICTAMINADOR



29410027 - SULLA TORRES RAUL RICARDO
DICTAMINADOR



29565731 - COPA PINEDA JUAN CARLOS
DICTAMINADOR



IDENTIFICACIÓN DE BILLETES FALSIFICADOS PERUANOS UTILIZANDO ARQUITECTURAS HÍBRIDAS DE APRENDIZAJE PROFUNDO CON CNN Y RNN APLICADAS A VISIÓN POR COMPUTADORA

INFORME DE ORIGINALIDAD

16%	14%	4%	5%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	repositorio.unap.edu.pe Fuente de Internet	3%
2	hdl.handle.net Fuente de Internet	3%
3	Submitted to Universidad Católica de Santa María Trabajo del estudiante	1%
4	repositorio.ucsm.edu.pe Fuente de Internet	<1%
5	Submitted to University of Brighton Trabajo del estudiante	<1%
6	Submitted to Universidad Internacional de la Rioja Trabajo del estudiante	<1%
7	www.campustudionline.com Fuente de Internet	<1%
8	Submitted to Universitat Oberta de Catalunya Trabajo del estudiante	<1%
9	www.coursehero.com Fuente de Internet	<1%
10	1library.co Fuente de Internet	<1%
11	sedici.unlp.edu.ar Fuente de Internet	<1%
12	sciencepg.org Fuente de Internet	<1%
13	repositorio.esan.edu.pe Fuente de Internet	<1%
14	Submitted to University of Alabama at Birmingham Trabajo del estudiante	<1%
	es.mathworks.com	

Dedicatoria

A Dios, por ser mi guía y fortaleza en cada momento de este camino. Por darme la sabiduría necesaria para enfrentar los desafíos y la perseverancia para no rendirme cuando el camino se hizo difícil.

A mi madre, por ser el pilar más grande de mi vida. Su esfuerzo incansable, su trabajo constante y su amor incondicional han sido el motor que me impulsó a llegar hasta aquí. Gracias por darme siempre más de lo que necesitaba, por motivarme a seguir adelante y por enseñarme con el ejemplo que todo esfuerzo tiene su recompensa. Este logro es tan tuyo como mío.

A mi hermana, quien estuvo presente en los primeros años de esta etapa universitaria. Gracias por la compañía, por los momentos compartidos y por ser parte de este proceso desde el inicio.

Agradecimientos

En primer lugar, agradezco a Dios por guiar mis pasos y permitirme culminar esta etapa con éxito.

A mi madre, quien con su trabajo y dedicación hizo posible mi formación profesional. Su apoyo siempre fue más allá de lo necesario, brindándome motivación y ejemplo de vida en cada momento.

A mi hermana, quien compartió conmigo los primeros años de esta etapa universitaria. Su compañía durante ese tiempo fue un soporte importante mientras me adaptaba y crecía en la carrera.

A mi asesor de tesis, por su disposición y paciencia para despejar mis dudas a lo largo del desarrollo de esta investigación. Su orientación fue clave para culminar este trabajo.

Finalmente, a los docentes de la Escuela Profesional de Ingeniería Electrónica de la Universidad Católica de Santa María, por los conocimientos y experiencias compartidas durante mi formación.

RESUMEN

La presente investigación tuvo como objetivo identificar billetes peruanos mediante arquitecturas de aprendizaje profundo basadas en CNN y CNN-RNN aplicadas a visión por computadora, con el fin de evaluar su desempeño en la clasificación de billetes verdaderos y falsificados según sus características visuales. El estudio se desarrolló bajo un enfoque cuantitativo, de tipo aplicado y mediante experimentación computacional, empleando imágenes de billetes peruanos procesadas con técnicas de visión artificial. En la etapa inicial, se analizaron las características visuales de los billetes mediante procesamiento digital de imágenes, considerando escala de grises, detección de bordes, gradientes, filtros Laplacianos y análisis frecuencial. Estas técnicas permitieron observar diferencias en textura, contraste, nitidez, bordes, iluminación y distribución de patrones visuales entre billetes auténticos y falsificados. Asimismo, se empleó aumentación de datos solo en el conjunto de entrenamiento, aplicando rotación, zoom, desplazamientos, variación de brillo y volteo horizontal. Posteriormente, se entrenaron tres arquitecturas: CNN, CNN-LSTM y CNN-GRU. La CNN permitió extraer patrones espaciales de las imágenes, mientras que CNN-LSTM y CNN-GRU incorporaron unidades recurrentes para complementar la clasificación. La evaluación se realizó en el conjunto de prueba mediante exactitud, precisión, recall, F1-score, matriz de confusión y AUC. Los resultados mostraron que la CNN alcanzó una exactitud de 98.22%, precisión para la clase falso de 100.00%, recall de 96.67%, F1-score de 98.31% y AUC de 0.9995. La CNN-LSTM obtuvo 98.04% de exactitud, 100.00% de precisión, 96.33% de recall, 98.13% de F1-score y AUC de 0.9997. Finalmente, la CNN-GRU presentó el mejor desempeño general, con 98.93% de exactitud, 99.66% de precisión, 98.33% de recall, 98.99% de F1-score y AUC de 0.9993. En consecuencia, se concluye que CNN-GRU fue el modelo con mejor comportamiento para identificar billetes peruanos falsificados.

Palabras clave: Aprendizaje profundo, Billetes falsificados, Visión por computadora

ABSTRACT

The objective of this research was to identify Peruvian banknotes using deep learning architectures based on CNNs and CNN-RNNs applied to computer vision, in order to evaluate their performance in classifying genuine and counterfeit banknotes based on their visual characteristics. The study was conducted using a quantitative, applied approach and computational experimentation, employing images of Peruvian banknotes processed with computer vision techniques. In the initial stage, the visual characteristics of the banknotes were analyzed using digital image processing, considering grayscale, edge detection, gradients, Laplacian filters, and frequency analysis. These techniques allowed for the observation of differences in texture, contrast, sharpness, edges, lighting, and the distribution of visual patterns between authentic and counterfeit banknotes. Additionally, data augmentation was applied only to the training set, using rotation, zooming, translations, brightness variation, and horizontal flipping. Subsequently, three architectures were trained: CNN, CNN-LSTM, and CNN-GRU. The CNN enabled the extraction of spatial patterns from the images, while CNN-LSTM and CNN-GRU incorporated recurrent units to enhance classification performance. The evaluation was performed on the test set using accuracy, precision, recall, F1-score, confusion matrix, and AUC. The results showed that the CNN achieved an accuracy of 98.22%, precision for the false class of 100.00%, recall of 96.67%, an F1-score of 98.31%, and an AUC of 0.9995. The CNN-LSTM achieved 98.04% accuracy, 100.00% precision, 96.33% recall, 98.13% F1-score, and an AUC of 0.9997. Finally, the CNN-GRU model demonstrated the best overall performance, with 98.93% accuracy, 99.66% precision, 98.33% recall, 98.99% F1-score, and an AUC of 0.9993. Consequently, it is concluded that the CNN-GRU model performed best in identifying counterfeit Peruvian banknotes.

Keywords: Deep learning, Counterfeit bills, Computer vision.

ÍNDICE

Dedicatoria

Agradecimientos

RESUMEN

ABSTRACT

INTRODUCCIÓN....1

CAPÍTULO I....2

PLANTEAMIENTO DE ESTUDIO....2

1.1 Identificación del Problema....3

1.2 Descripción del problema....4

1.3 Objetivos....4

1.3.1 Objetivo General....4

1.3.2 Objetivos Específicos....4

1.4 Variables....4

1.5 Hipótesis....5

1.5.1 Hipótesis General....5

1.5.2 Hipótesis Específicas....5

1.6 Alcances....5

CAPÍTULO II....7

MARCO TEÓRICO....7

2.1 Antecedentes de la investigación....8

2.1.1 Antecedentes Nacionales....8

2.1.2 Antecedentes Internacionales....9

2.2 Bases Teóricas....10

2.2.1 Visión por computadora....10

2.2.2 Billetes falsificados....23

2.2.3 Características visuales de los billetes....26

2.2.4 Convolutional Neural Network (Red Neuronal Convolucional)....31

2.2.5 Recurrente Neural Network (Red Neuronal Recurrente)....34

CAPÍTULO III....40

PLANTEAMIENTO DE LA INVESTIGACIÓN....40

3.1 Metodología de Investigación....41

3.2 Tipo de Investigación....41

3.3 Diseño de Investigación....41

3.4 Técnicas e Instrumentos 41
3.4.1 Técnicas 41
3.4.2 Instrumentos 42
3.5 Campo de Verificación 42
3.5.1 Ubicación Temporal 42
3.5.2 Ubicación espacial 42
3.5.3 Unidades de Estudio 42
CAPÍTULO IV 44
DISEÑO E IMPLEMENTACIÓN DE LA ARQUITECTURA HÍBRIDA DE APRENDIZAJE PROFUNDO 44
4.1. Dataset de billetes peruanos 45
4.2. Características visuales de los billetes peruanos 46
4.3. Arquitecturas CNN y CNN-RNN 67
4.4. Entrenamiento de modelos 74
4.4.1. Análisis de características visuales con PDI 76
4.4.2. Generadores de datos (aumentación solo en train) 80
4.4.3. Entrenamiento de los tres modelos CNN, CNN-LSTM y CNN-GRU 85
4.4.4. Evaluación en el conjunto de Test 90
CAPÍTULO V 96
RESULTADOS Y ANÁLISIS 96
5.1. Evaluación de modelos CNN y RNN 96
5.1.1. Prueba para un billete de 100 soles – enfoque verdadero 97
5.1.2. Prueba para un billete de 100 soles – enfoque falso 101
CONCLUSIONES 107
RECOMENDACIONES 109
REFERENCIAS BIBLIOGRÁFICAS 110

ÍNDICE DE TABLAS

Tabla 1 *Clasificación de dispositivos de captura para visión por computadora* 15
Tabla 2 *Etapas del procesamiento en una Red Neuronal Convolucional (CNN)* 32
Tabla 3 *Arquitectura 1: CNN Puro* 69
Tabla 4 *Arquitectura 2: CNN-LSTM Bidireccional* 70
Tabla 5 *Arquitectura 3: CNN-GRU Bidireccional* 72
Tabla 6 *Comparativa de parámetros - Arquitecturas de Red Neuronal* 73
Tabla 7 *Desempeño por modelo en prueba* 94
Tabla 8 *Comparación general de modelos en prueba* 95

ÍNDICE DE FIGURAS

Figura 1 *Proceso general de análisis de imágenes en visión por computadora* 11
Figura 2 *Etapas de un sistema de visión por computadora* 12
Figura 3 *Esquema del sistema de visión por computadora* 13
Figura 4 *Esquema de un filtro digital* 16
Figura 5 *Ejemplo de segmentación para identificar texto* 16
Figura 6 *Esquema del sistema de clasificación de patrones* 17
Figura 7 *Representación del par estereoscópico* 18
Figura 8 *Perspectiva a dos puntos en una escena tridimensional* 19
Figura 9 *Trayectoria de rayos en un sistema óptico* 19
Figura 10 *Patrón de filtro de color Bayer* 20
Figura 11 *Procesamiento de imágenes* 21
Figura 12 *Detección y coincidencia de características* 22
Figura 13 *Segmentación de imágenes* 23
Figura 14 *Billete falso* 24
Figura 15 *Elementos de seguridad del billete de 10 soles* 27
Figura 16 *Elementos de seguridad del billete de 20 soles* 28
Figura 17 *Elementos de seguridad del billete de 50 soles* 29
Figura 18 *Elementos de seguridad del billete de 100 soles* 30
Figura 19 *Arquitectura general de una red neuronal convolucional (CNN)* 32
Figura 20 *Red neuronal pre-alimentada* 34
Figura 21 *Esquema de conexiones recurrentes en una RNN* 35
Figura 22 *Estructura de conexiones en una Red Hopfield* 35
Figura 23 *Diagrama de red parcialmente recurrente* 36
Figura 24 *Esquema de una red totalmente recurrente* 37
Figura 25 *Redes Neuronales Recurrentes Básicas* 38
Figura 26 *Redes Neuronales Recurrentes Bidireccionales* 39
Figura 27 *Análisis de espacios de color para clasificación de billetes* 47
Figura 28 *Características para la detección de falsificaciones* 50
Figura 29 *Filtros de suavizado - mejora de ruido* 53
Figura 30 *Mejora de contraste - realce de detalles* 57
Figura 31 *Detección de bordes* 60
Figura 32 *Operaciones morfológicas - análisis de forma* 66

Figura 33 *Preparación de datos para entrenamiento* 75
Figura 34 *Comparación de técnicas de procesamiento digital para la detección de billetes verdaderos y falsos* 79
Figura 35 *Aumento de datos para entrenamiento del modelo* 81
Figura 36 *Muestra de imágenes clasificadas por el modelo verdadero o falso* 84
Figura 37 *Proceso de entrenamiento del modelo CNN* 86
Figura 38 *Entrenamiento del modelo CNN-LSTM* 87
Figura 39 *Entrenamiento del modelo CNN-GRU* 88
Figura 40 *Curvas de entrenamiento y validación de los modelos CNN, CNN-LSTM y CNN-GRU* 89
Figura 41 *Matrices de confusión de los modelos evaluados* 91
Figura 42 *Curvas ROC y comparación de métricas de los modelos evaluados* 93
Figura 43 *Función de predicción de billetes mediante el mejor modelo entrenado* 97
Figura 44 *Reconocimiento de captura de un Billete verdadero de 100 soles* 98
Figura 45 *Predicción por condición y modelo de un Billete verdadero de 100 soles* 99
Figura 46 *Robustez ante condiciones de captura de un Billete verdadero de 100 soles* 100
Figura 47 *Reconocimiento de captura de un Billete falso de 100 soles* 102
Figura 48 *Predicción por condición y modelo de un Billete falso de 100 soles* 103
Figura 49 *Robustez ante condiciones de captura de un Billete falso de 100 soles* 105

ÍNDICE DE ANEXOS

Anexo N° 1. Cuadro de Operacionalización de variables 115
Anexo N° 2. Código del sistema de identificación de billetes falsificados.....116

INTRODUCCIÓN

La circulación de billetes falsificados continúa representando un problema para las transacciones económicas realizadas en el Perú, especialmente en comercios minoristas, transporte y actividades donde la verificación del dinero se efectúa de forma visual. Aunque el Banco Central de Reserva del Perú ha incorporado elementos de seguridad avanzados en las nuevas familias de billetes, todavía se detectan unidades adulteradas que son identificadas durante operativos e inspecciones bancarias. Esta situación afecta a usuarios y establecimientos, ya que los billetes falsos carecen de valor legal y generan pérdidas económicas directas. La verificación manual basada en observación puede resultar insuficiente cuando existen similitudes en textura, color o impresión entre billetes legítimos y falsificados, o cuando el billete presenta deterioro por uso. Por este motivo, se han desarrollado sistemas de identificación automatizada que utilizan visión por computadora para analizar características visuales asociadas a los elementos de seguridad del papel moneda. El uso de aprendizaje profundo permite extraer y procesar patrones complejos presentes en las imágenes capturadas, lo que incrementa la precisión en la clasificación.

En este contexto, la investigación propone el desarrollo de un modelo híbrido compuesto por redes neuronales convolucionales (CNN) y redes neuronales recurrentes (RNN) aplicado a la identificación de billetes peruanos falsificados. La CNN se utilizará para extraer características espaciales de la imagen, mientras que la RNN procesará información secuencial relacionada con variaciones visuales del billete. El objetivo es evaluar si la arquitectura CNN–RNN mejora la detección en comparación con modelos individuales.

CAPÍTULO I

PLANTEAMIENTO DE ESTUDIO

1.1 Identificación del Problema

A nivel internacional, la falsificación de billetes continúa siendo una actividad delictiva que afecta la confianza en los medios de pago físicos. En 2023 se retiraron de circulación aproximadamente 467 000 billetes de euro falsos en la zona euro, lo que equivale a 16 billetes falsos por cada millón de billetes auténticos en circulación; las denominaciones de 20 y 50 euros concentraron más del 70 % de los casos detectados (European Central Bank, 2024). En distintos países se han registrado incrementos puntuales de falsificaciones en monedas locales, afectando especialmente al comercio minorista, transporte y establecimientos que manejan efectivo sin equipos de verificación avanzados. Los reportes internacionales coinciden en que la detección visual del usuario suele fallar cuando el billete presenta desgaste, suciedad o variaciones de impresión (Antonius et al., 2023).

A nivel nacional, el Banco Central de Reserva del Perú (BCRP) reporta que, aunque la retención de billetes falsos se ha reducido en aprox. 85% desde 2009, se siguen detectando miles de unidades cada año en operativos y depósitos bancarios. Informes públicos señalan que los billetes de 20 y 100 soles son los más falsificados, afectando principalmente a mercados, transporte urbano, bodegas y otros puntos donde la revisión se realiza de manera manual (G4s, 2022). Esta problemática se observa en comercios pequeños, ferias, restaurantes y servicios cotidianos donde la revisión del efectivo se basa únicamente en la observación del usuario, lo que incrementa el riesgo de circulación de billetes adulterados. Además, los casos reportados por la Policía Nacional del Perú y por el BCRP muestran que las organizaciones dedicadas a la falsificación utilizan técnicas de impresión que dificultan distinguir a simple vista un billete legítimo de uno adulterado, especialmente cuando el billete presenta uso o deterioro.

En el Perú continúa circulando una cantidad significativa de billetes falsificados, especialmente en comercios pequeños, transporte y mercados donde la verificación se realiza de manera visual y rápida. Aunque el BCRP ha mejorado los elementos de seguridad y publica informes sobre falsificaciones, todavía se detectan miles de billetes adulterados cada año. Además, los métodos de revisión manual no siempre permiten identificar variaciones en textura, color, seriales o impresiones irregulares, sobre todo cuando los billetes están desgastados.

1.2 Descripción del problema

En el Perú continúan detectándose billetes falsificados en transacciones cotidianas, principalmente en mercados, transporte y comercios donde la verificación se hace de forma visual. Los billetes falsos suelen pasar desapercibidos cuando presentan colores, impresiones o texturas similares a los originales, especialmente si el billete está desgastado. Esta situación genera pérdidas económicas para los usuarios y evidencia dificultades para identificar falsificaciones en condiciones reales de uso.

1.3 Objetivos

1.3.1 Objetivo General

Identificar billetes peruanos falsificados mediante arquitecturas de aprendizaje profundo basadas en CNN y CNN-RNN aplicadas a visión por computadora, a fin de evaluar su desempeño en la detección de falsificaciones según sus características visuales.

1.3.2 Objetivos Específicos

a) Analizar las características visuales de los billetes peruanos para definir las entradas que serán procesadas por la CNN.
b) Entrenar una red CNN para extraer patrones espaciales relacionados con billetes falsos y auténticos.
c) Implementar una red RNN que procese información secuencial del billete para complementar la clasificación.
d) Evaluar el desempeño del modelo híbrido CNN-RNN en diferentes condiciones de captura y compararlo con modelos individuales.

1.4 Variables

Variable Independiente: Arquitectura híbrida de aprendizaje profundo CNN–RNN

Indicadores:

- Número de capas utilizadas en la CNN y en la RNN
- Exactitud del modelo en el entrenamiento (training accuracy)
- Pérdida del modelo durante el entrenamiento (training loss)

- Cantidad de características extraídas por la CNN

Variable Dependiente: Identificación de billetes peruanos falsificados

Indicadores:

- Precisión en la detección de billetes falsos
- Exactitud global del modelo
- Sensibilidad para billetes falsos
- Tasa de falsos positivos
- Tasa de falsos negativos
- Matriz de confusión obtenida en la clasificación final

1.5 Hipótesis

1.5.1 Hipótesis General

La aplicación de arquitecturas de aprendizaje profundo basadas en CNN y CNN-RNN, en el marco de la visión por computadora, influye significativamente en la identificación de billetes peruanos falsificados a partir de sus características visuales.

1.5.2 Hipótesis Específicas

a) Las características visuales extraídas de los billetes permiten a la CNN identificar patrones espaciales asociados a billetes falsos y auténticos.
b) El entrenamiento de la CNN contribuye a mejorar la detección de diferencias visuales entre billetes legítimos y adulterados.
c) El procesamiento secuencial realizado por la RNN incrementa la precisión al analizar secuencias del billete.
d) El modelo híbrido CNN-RNN obtiene un desempeño superior bajo diferentes condiciones de captura en comparación con las redes utilizadas de manera independiente.

1.6 Alcances

La investigación se limita al análisis y evaluación de billetes peruanos de distintas denominaciones mediante técnicas de visión por computadora. El estudio se desarrolla en un

entorno controlado usando imágenes capturadas bajo diferentes condiciones de iluminación, ángulos y niveles de desgaste. El trabajo se enfoca exclusivamente en el diseño, entrenamiento y validación de un modelo híbrido CNN–RNN Convolutional Neural Network (Red Neuronal Convolucional) y Recurrent Neural Network (Red Neuronal Recurrente) para la identificación de billetes falsificados, sin incluir el desarrollo de hardware ni la implementación en dispositivos físicos. El análisis se centra en comparar el desempeño del modelo híbrido con modelos individuales utilizando métricas de clasificación. Los resultados representan el comportamiento del sistema dentro del conjunto de datos empleado y no abarcan la detección de otros tipos de documentos o monedas distintas al sol peruano.

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes de la investigación

2.1.1 Antecedentes Nacionales

Quispe y Ching (2024) desarrollaron un modelo predictivo capaz de identificar fibras textiles naturales utilizando imágenes microscópicas, con el fin de mejorar la autenticidad de productos artesanales y apoyar a productores y artesanos en el sector textil altoandino. La metodología aplicada siguió el enfoque CRISP-DM, que incluyó la captura estandarizada de imágenes, la construcción de un conjunto de datos especializado, el preprocesamiento mediante aumentación y la evaluación de diversas arquitecturas de redes neuronales convolucionales bajo técnicas de aprendizaje profundo. Los resultados mostraron que el modelo optimizado, basado en arquitecturas CNN modernas, alcanzó una alta precisión de clasificación y demostró capacidad de generalizar adecuadamente.

Casanova y Bazán (2024) desarrollaron un modelo de visión artificial basado en Deep Learning capaz de detectar actividades delictivas con pistolas en tiempo real dentro de Lima Metropolitana. Para lograrlo, se empleó una metodología que combinó la creación de un dataset propio obtenido desde el videojuego Grand Theft Auto V, el uso del algoritmo YOLO para la detección de objetos y un proceso iterativo de desarrollo bajo el marco Scrum. La metodología incluyó la recopilación, limpieza y etiquetado de imágenes, el entrenamiento del modelo y la implementación de una aplicación de videovigilancia. Como resultado, el sistema alcanzó una alta precisión en la identificación de armas de fuego, permitiendo generar alertas automáticas y mejorando el soporte a los operadores de cámaras.

Salinas (2024) identifico el mejor modelo predictivo para detectar fraude en mensajes de texto en el contexto peruano. Para ello, se desarrolló una metodología que incluyó la recolección de 527 imágenes, de las cuales se extrajeron 1740 mensajes que luego fueron etiquetados manualmente como FRAUDE o LEGÍTIMO. Posteriormente, estos registros se integraron con bases adicionales, alcanzando un total de 4475 mensajes. La metodología aplicada comprendió el preprocesamiento del texto, la vectorización con modelos Word2Vec y FastText, y el entrenamiento de modelos basados en Redes Neuronales Recurrentes, incluyendo RNN, LSTM, GRU y combinaciones con CNN. Los resultados mostraron que el mejor desempeño fue obtenido por una arquitectura RNN de tres capas utilizando el embedding FastText-NewL, logrando una precisión del 85.62%.

2.1.2 Antecedentes Internacionales

Pachón et al. (2023) desarrollaron un modelo de aprendizaje profundo más eficiente para el reconocimiento de billetes falsificados, aplicando una metodología de poda que reduce drásticamente los parámetros de redes CNN sin afectar significativamente su precisión. La metodología utilizada consistió en un proceso de pruning filter-wise, aplicado en capas convolucionales y totalmente conectadas, empleando la norma L2 como criterio de eliminación. Este método también evaluó el impacto cuantitativo y cualitativo mediante métricas como parámetros, FLOPs y precisión, además de interpretabilidad con HiResCAM. Los resultados demostraron reducciones de hasta 95% en parámetros y FLOPs, manteniendo pérdidas menores al 1% de exactitud en modelos como VGG11 y el modelo personalizado, evidenciando alta eficiencia computacional sin comprometer el desempeño.

Hamed y Azzo (2024) desarrollaron un sistema capaz de identificar billetes falsificados mediante técnicas avanzadas de machine learning y deep learning, analizando las características de seguridad de los dinares iraquíes. La metodología empleó un conjunto de modelos de aprendizaje profundo como CNN, VGG16, MobileNetV2 e InceptionV3 complementados con algoritmos de machine learning como Random Forest, XGBoost, CatBoost y SVM. El proceso incluyó extracción de características, transferencia de aprendizaje y entrenamiento supervisado con imágenes de billetes reales y falsos. Los resultados demostraron altos niveles de precisión, destacando las arquitecturas CNN, CatBoost y SVM, que alcanzaron más del 98% de exactitud, confirmando la eficacia de estas técnicas para la detección confiable de billetes falsificados.

Sadyk et al. (2024) analizaron cómo los métodos de deep learning permiten mejorar la detección de billetes falsos frente a técnicas tradicionales. La metodología empleada consistió en una revisión sistemática del estado del arte, comparando modelos avanzados como CNN, RNN y GAN con enfoques clásicos de reconocimiento y validación de billetes. El análisis incluyó la evaluación de arquitecturas, procesos de aprendizaje, extracción de características y casos de aplicación en distintos países y monedas. Los resultados mostraron que las técnicas de deep learning ofrecen mayor precisión, mejor capacidad de adaptación y menor intervención humana, logrando identificar patrones complejos que los métodos convencionales no detectan.

2.2 Bases Teóricas

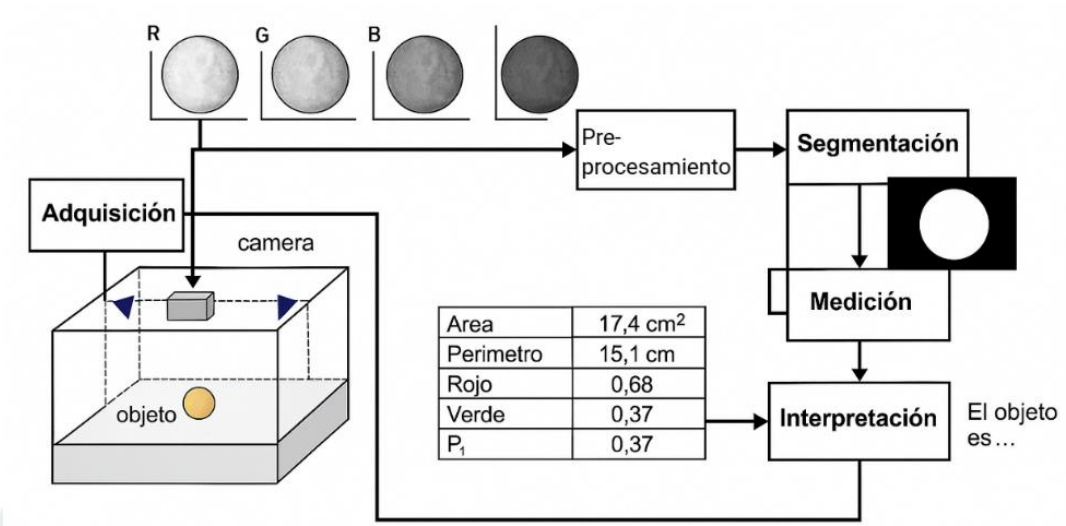
2.2.1 Visión por computadora

La visión por computadora se entiende actualmente como el tratamiento de imágenes mediante la aplicación de filtros digitales que dividen la información y producen nuevas versiones de la imagen. El propósito de este concepto es entregar una imagen orientada a un análisis concreto, es decir, la escena se separa según lo que se busca resaltar o identificar en la imagen procesada. En tiempos recientes, se han desarrollado programas basados en visión automatizada para medir, interpretar y tomar decisiones en procesos de control de calidad de materiales, recuperación y mejora de imágenes con interferencias, así como para la detección y clasificación de escenas (Maguiña y Pascacio, 2021).

La visión por computador se entiende como un área científica orientada a que los sistemas informáticos puedan analizar y comprender información proveniente de imágenes y secuencias de video, de forma semejante al proceso visual humano. Su propósito es que un equipo logre identificar lo que aparece en una escena y utilice esa información para ejecutar acciones o apoyar la toma de decisiones. Este procedimiento abarca la adquisición de imágenes, la mejora de su calidad, la detección de rasgos relevantes y el análisis de los datos obtenidos para su uso en distintos entornos. Además, se reconoce que la visión computacional integra aportes de disciplinas como matemáticas, ingeniería y ciencias de la computación, lo que permite su aplicación en soluciones actuales relacionadas con identificación de personas, vehículos autónomos o sistemas de realidad aumentada. Su capacidad para automatizar la interpretación de información visual resulta importante en tareas de investigación y en diversos sectores productivos (Vega, 2023).

Figura 1

Proceso general de análisis de imágenes en visión por computadora



Nota. Elaboración propia.

Vélez et al. (2003), citados por Loaiza et al. (2012), señalan que la visión por computadora tiene como propósito obtener información del entorno real a partir de imágenes mediante el uso de un computador. Mencionan que este objetivo es amplio y de alta complejidad, y que en la actualidad aún se encuentra en una fase inicial (p.16).

Por otra parte, Szeliski (2010) indica que la visión por computadora tiene un enfoque similar al humano orientado a la percepción, ya que somos capaces de interpretar la estructura tridimensional del entorno en el que vivimos con gran facilidad. Asimismo, presenta un ejemplo cotidiano que muestra cómo podemos identificar rasgos complejos como la forma y la transparencia de los pétalos en un sencillo florero, e incluso distinguir ligeras variaciones de luz y sombra. De igual modo, al observar un retrato grupal enmarcado, es posible contar y reconocer a las personas presentes en la imagen e inferir sus emociones a partir de sus expresiones faciales.

Porras (2014) señala que la función central de la visión por computadora consiste en imitar el mecanismo de la vista en los seres vivos. Esto implica que, cuando un objeto es captado por los sensores, la información obtenida se transmite como señales que, posteriormente, serán procesadas por un sistema encargado de interpretar dichos datos, de forma similar a como lo hace el cerebro.

Respecto al control de procesos en tiempo real, las técnicas de visión por computadora permiten obtener datos de forma rápida. Esto implica que los procesos pueden operar de manera autónoma, verificando de forma detallada cada componente fabricado (Alegre et al., 2008).

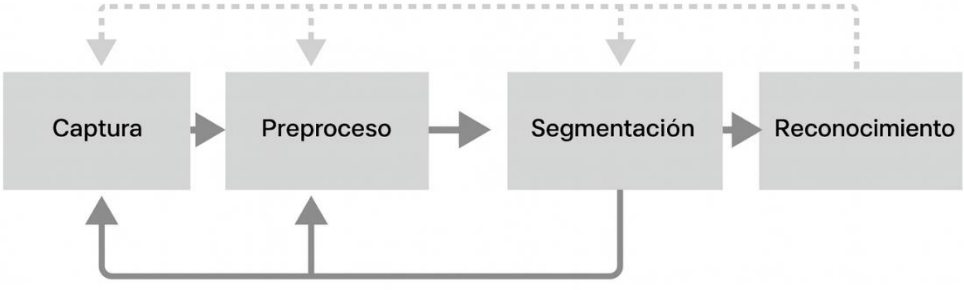
Asimismo, se señalan dos expresiones equivalentes: “la visión artificial o visión por computadora es el área científica y tecnológica que permite a las máquinas observar, obtener datos de imágenes digitales, llevar a cabo alguna tarea o interpretar la escena que están observando”. A partir de las definiciones mencionadas en los párrafos anteriores, se reconoce de manera general que la visión artificial se centra en la obtención de información a partir de imágenes y, en consecuencia, busca reproducir el sentido de la vista de los seres vivos, ejecutando dicho proceso de forma rápida (García, 2012).

2.2.1.1 Etapas de la Visión por computadora

Vélez et al. (2003) indican que “el ser humano recibe la luz mediante los ojos, y que dicha información avanza por el nervio óptico hasta llegar al cerebro, donde es tratada” (p.18). Del mismo modo, mencionan que la división de las imágenes implica localizar componentes más básicos, representando la primera fase del procesamiento; luego, el cerebro interpreta la escena y, finalmente, realiza una acción. Por ello, con el objetivo de reproducir este mecanismo, Vélez et al. (2003) señalan cuatro etapas principales, como se muestra en la figura 2.

Figura 2

Etapas de un sistema de visión por computadora



Nota. Tomado de (Vélez et al., 2003).

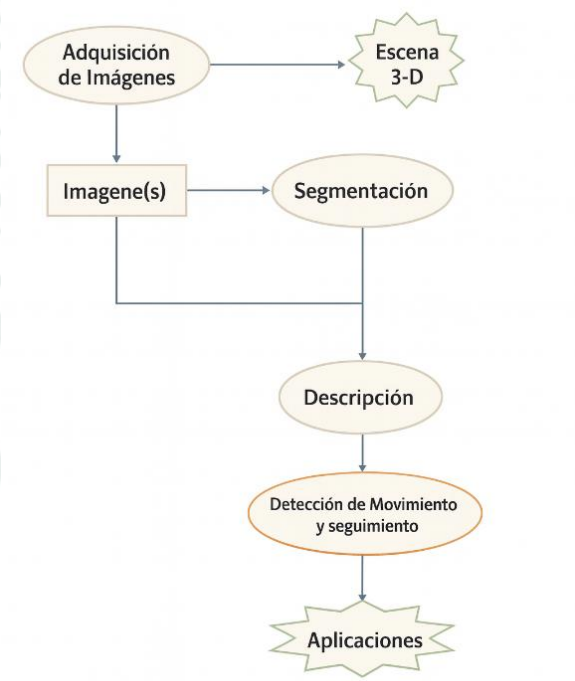
La primera etapa, de carácter sensorial, está relacionada con la obtención de imágenes digitales mediante algún tipo de sensor. La segunda fase se orienta al procesamiento digital de

las imágenes con el objetivo de facilitar las etapas siguientes; en este preprocesamiento, por medio de filtros y transformaciones geométricas, se eliminan elementos no deseados de la imagen o se resaltan zonas de interés. La fase posterior se conoce como segmentación y consiste en separar los componentes relevantes de la escena para su análisis. Por último, en la etapa de reconocimiento o clasificación, se busca identificar los objetos segmentados mediante el estudio de determinadas características diferenciadoras definidas previamente (Vélez et al., 2003, pp. 19-20).

Por otro lado, Loaiza et al. (2012) presentan en un esquema el procedimiento general de visión artificial (figura 3), el cual incluye la captura de imágenes, la segmentación, la descripción y la identificación del movimiento y su seguimiento.

Figura 3

Esquema del sistema de visión por computadora



Nota. Tomado de (Loaiza et al., 2012).

De acuerdo con el esquema mostrado en la figura 3, Loaiza et al. (2012) explican los siguientes procedimientos: El primero se denomina adquisición de imágenes, el cual abarca la obtención de imágenes de intensidad, de alcance y térmicas, obteniéndose en todos los casos una matriz bidimensional de valores. El segundo corresponde a la segmentación, que implica dividir una imagen digital en áreas similares basadas en características determinadas, con el

objetivo de facilitar su análisis. El tercer proceso, denominado descripción, emplea las zonas de interés para extraer propiedades o atributos que permitan reconocer estructuras necesarias para su aplicación posterior. Finalmente, la detección de movimiento y seguimiento comprende actividades destinadas a diferenciar entre el fondo de la escena y los objetos en desplazamiento; identificar dichos objetos, ya sea como una persona o como el elemento de interés; y, por último, rastrear el movimiento de los mismos a lo largo de la escena (pp. 89-91).

2.2.1.2 Clasificación de Visión por computadora

La visión por computadora es organizada por Vélez et al. (2003) en cinco etapas, mientras que Szeliski (2010) plantea trece procedimientos necesarios que deben considerarse. Ambas perspectivas se muestran a continuación:

a) Clasificación según Vélez

Por una parte, Vélez et al. (2003) divide la visión artificial en cinco procesos:

- **Adquisición y representación de imágenes digitales**

  Esta etapa reúne los aspectos principales de la fase que comprende la captura y conversión de una imagen, es decir, la obtención desde el entorno físico y su paso al ámbito discreto y digital. Vélez et al. (2003) mencionan que una imagen digital bidimensional está integrada por un conjunto de píxeles, y cada píxel aporta información sobre un área específica de la imagen. En imágenes en escala de grises, dicha información está asociada al nivel de brillo; mientras que, en imágenes a color, puede obtenerse mediante la intensidad individual de los componentes presentes en el modelo cromático, como RGB. Además, en este proceso intervienen métodos de compresión orientados a encontrar la forma más eficiente de almacenar imágenes. Finalmente, esta fase también considera las relaciones básicas entre los píxeles que pueden establecerse dentro de una imagen. (Vélez et al., 2003, p. 23).

  Esto permite concluir que la selección del equipo de captura vinculado al computador (figura 4), así como el propósito asignado, tiene una influencia considerable en el resultado de este nivel.

Tabla 1

Clasificación de dispositivos de captura para visión por computadora

Categoría principal	Subcategoría	Elementos / Descripción
Escáner 3D	-	• Captura tridimensional de escenas u objetos.
Software de generación de imágenes sintéticas	-	• Imágenes creadas por simulación o renderizado por computadora.
Cámaras de Video	Analógicas	• Requieren Conversor Analógico Digital para conexión al ordenador.
	Digitales (CCD)	• Disco duro portátil para almacenamiento directo. • Tarjeta de memoria para registro de imágenes o video.
Cámaras Fotográficas	Convencionales	• Utilizan Procesado químico. • Papel fotográfico (copia física). • Transparencia. • Negativo. • Diapositiva. Para digitalizar material físico se emplean: • Escáner de tambor • Escáner plano de película • FotoCD
Salida / Destino común	-	Ordenador / Computadora donde se procesan las imágenes capturadas.

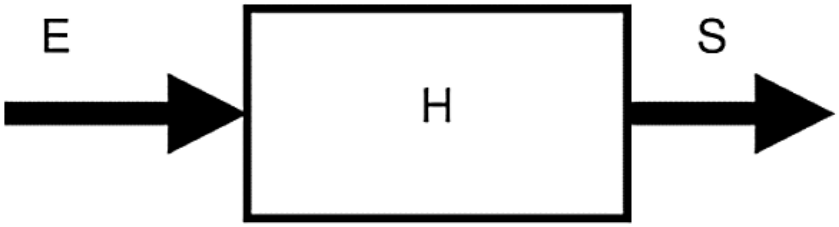
Nota. Tomado de (Vélez et al., 2003).

- **Filtrado y realzado de imagen**

De igual manera, para este nivel, Vélez et al. (2003) señala que dicho procedimiento comprende las operaciones y modificaciones aplicadas a las imágenes digitales antes de la etapa de segmentación y reconocimiento, con el objetivo de resaltar o mejorar algún elemento de la imagen, facilitando que las siguientes fases de segmentación y clasificación se lleven a cabo de manera más sencilla. Asimismo, la teoría de filtros describe parte de las acciones incluidas en este proceso; en ese sentido, un filtro puede entenderse como un tipo de modificación o ajuste aplicado a una señal de entrada, que mediante su transferencia produce una señal de salida, tal como se observa en la figura n.° 5. Estas señales suelen ser, por lo general, sonidos o imágenes (p. 61).

Figura 4

Esquema de un filtro digital



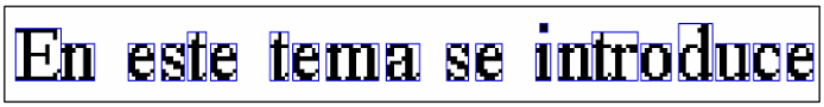
Nota. Tomado de (Vélez et al., 2003).

- **Segmentación**

Vélez et al. (2003) señalan que, en este procedimiento, se generan zonas uniformes a partir de una imagen digital, considerando propiedades como la intensidad del brillo o el color, con el objetivo de facilitar su análisis y posterior identificación. Un ejemplo de ello es la detección del rostro de una persona o la delimitación de un texto específico en una fotografía, como se muestra en la figura 6 (p. 111).

Figura 5

Ejemplo de segmentación para identificar texto



Nota. Tomado de (Vélez et al., 2003).

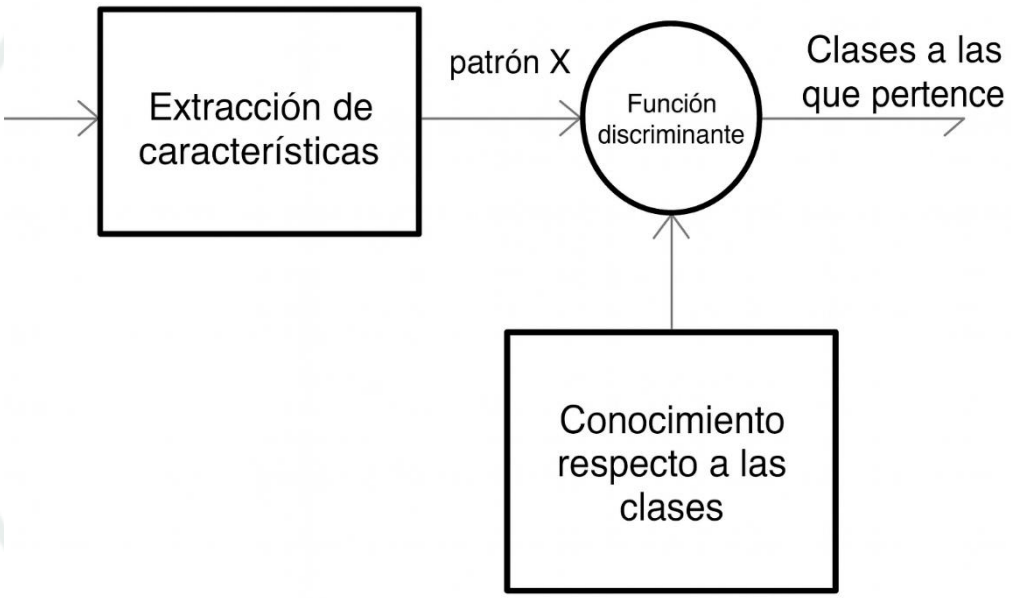
Aunque existen diversas formas de llevar a cabo este procedimiento, es posible desarrollar técnicas ajustadas al problema seleccionado, debido a que se rige por criterios definidos. Además, se contemplan métodos que permiten dividir los objetos sin depender de su ubicación o tamaño, con el propósito de superar algunas dificultades en la representación de los elementos obtenidos en este proceso (Vélez et al., 2003, citado por Loaiza et al., 2012, p. 90). Lo mencionado anteriormente expone un procedimiento útil para la labor de detección de matrículas vehiculares y rostros en el ámbito de la seguridad ciudadana, ya que posibilita sustituir el trabajo del observador humano, haciéndolo más eficiente y exacto.

- **Clasificadores**

Para entender este procedimiento, Vélez et al. (2003) utiliza el concepto de clases, las cuales, mediante diversos algoritmos, permiten organizar los elementos presentes en una escena con el fin de interpretarla. Dichos algoritmos se encargan de distinguir entre varios objetos dentro de un entorno de trabajo. Los clasificadores son técnicas que facilitan determinar de forma automática si ciertos objetos de un conjunto pertenecen a una clase específica (figura 7) (p. 151).

Figura 6

Esquema del sistema de clasificación de patrones



Nota. Tomado de (Vélez et al., 2003).

Con estos datos es posible establecer una relación para los fines de esta investigación: objeto (elemento identificado en la vía), universo de análisis (calles del distrito) y clases (vehículos, peatones, señales de tránsito, entre otros).

- **Visión tridimensional**

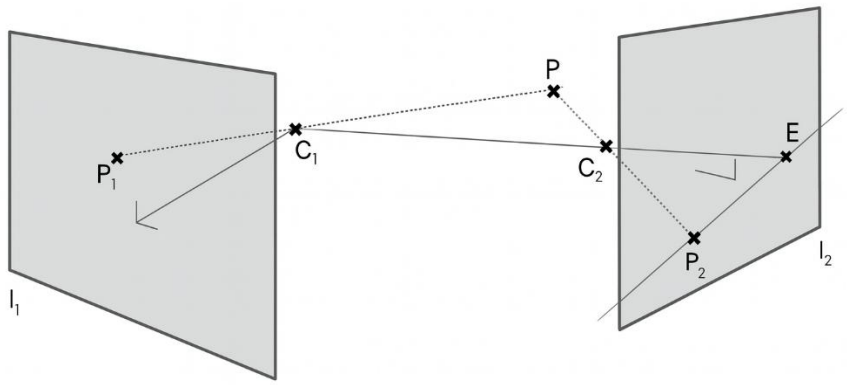
Vélez et al. (2003) señalan que obtener un sistema tridimensional de una escena representa uno de los retos más importantes. La reconstrucción en tres dimensiones para algunos seres vivos resulta tan natural como respirar, aunque

implica la proyección de un entorno real sobre áreas fotosensibles. Para plantear el problema de manera más sencilla, emplean el método del par estereoscópico mostrado en la figura 8 (p. 187).

Asimismo, Vélez et al. (2003) explican que la reconstrucción tridimensional requiere identificar correspondencias entre dos o más imágenes obtenidas desde distintas posiciones, lo cual permite estimar la profundidad de los objetos presentes en la escena. Este procedimiento se basa en principios geométricos que relacionan la posición de un punto en el espacio con sus proyecciones en los planos de imagen. A partir de estas correspondencias, es posible generar modelos 3D que describen la forma y ubicación de los elementos capturados, facilitando su interpretación en aplicaciones como la robótica, la inspección automatizada y los sistemas de ayuda a la navegación.

Figura 7

Representación del par estereoscópico

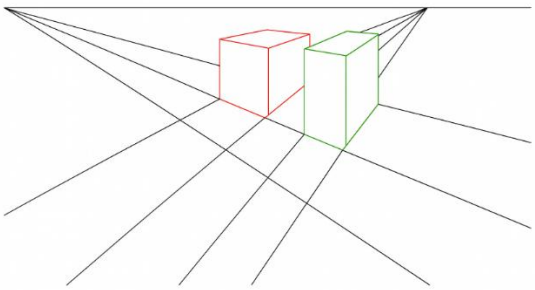


Nota. Tomado de (Vélez et al., 2003).

Esto puede observarse en el procedimiento de caza de un depredador que se desplaza por el aire, el cual logra distintas vistas de la presa a la que sigue mediante sus movimientos. De forma similar, las cámaras tipo domo pueden imitar esta función, ya que giran de manera continua para capturar imágenes desde diferentes ángulos.

Figura 8

Perspectiva a dos puntos en una escena tridimensional

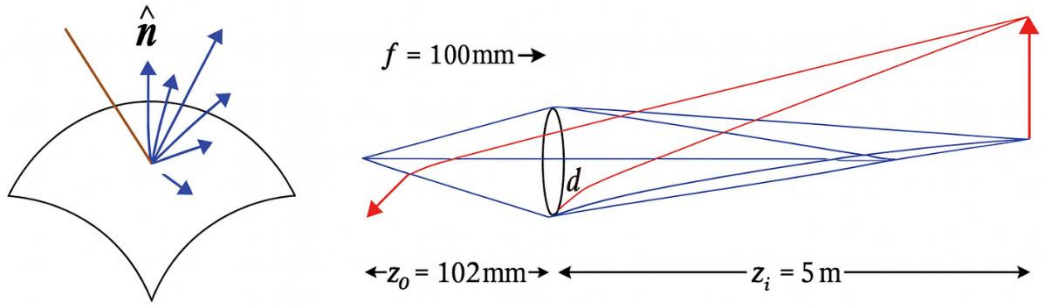


Nota. Tomado de (Szeliski, 2010).

Por otra parte, se explica cómo la luz, las características de la superficie (parte izquierda de la figura n.° 10) y la óptica de la cámara (parte derecha de la figura 10) se relacionan para generar los valores de color que son obtenidos por el sensor (p.31).

Figura 9

Trayectoria de rayos en un sistema óptico

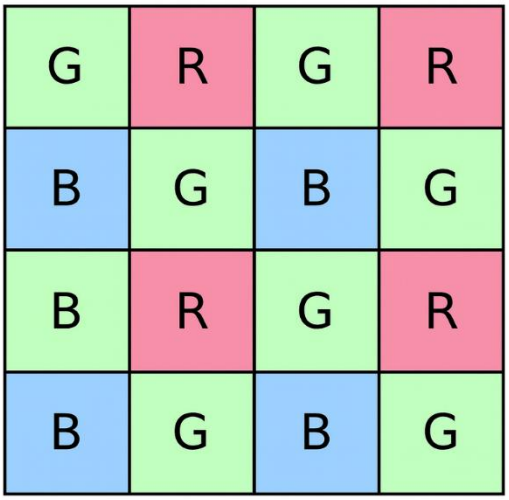


Nota. Tomado de (Szeliski, 2010).

Finalmente, se establece el procedimiento que implica transformar las imágenes con color continuo en muestras digitales discretas dentro del sensor de imagen (figura 11), con el fin de evitar problemas de muestreo, como el fenómeno de “aliasing”.

Figura 10

Patrón de filtro de color Bayer



Nota. Tomado de (Szeliski, 2010).

- **Procesamiento de imágenes**

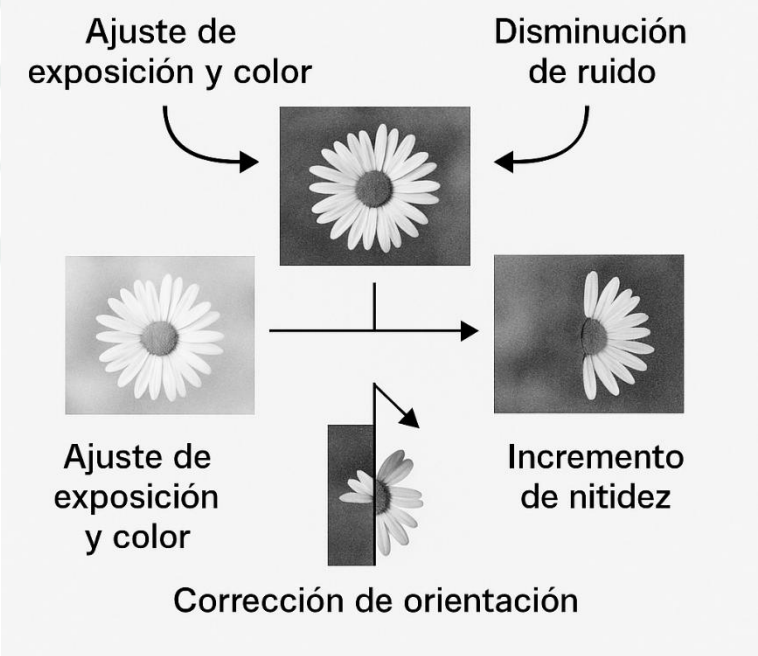
Teniendo en cuenta que las imágenes se generan mediante la interacción entre los elementos de una escena 3D, la iluminación y las ópticas y sensores de la cámara, es posible centrarse en la etapa inicial de la mayoría de aplicaciones en visión artificial. En otras palabras, este procedimiento permite realizar un preprocesamiento de la imagen y facilitar su análisis posterior. Para fines prácticos, se mencionan algunos ejemplos, como el ajuste de exposición y balance de color, la disminución de ruido en la imagen, el incremento de la nitidez o la corrección de orientación al girar la imagen. Para obtener resultados adecuados en esta fase, es importante considerar que la fotografía computacional y el reconocimiento incluyen el procesamiento de imágenes dentro del campo de la visión artificial (Szeliski, 2010, p. 101).

A partir de que una imagen se forma por la relación entre los objetos presentes en una escena tridimensional, las condiciones de iluminación y las características ópticas y del sensor de la cámara, se entiende que la primera fase de muchos sistemas de visión computacional se centra en el tratamiento inicial de dicha imagen. Esto permite preparar el contenido visual para que su interpretación posterior sea más sencilla y precisa. En la práctica, este tratamiento puede incluir tareas como ajustar la iluminación y el color, reducir interferencias o ruido, mejorar

la claridad de los bordes o modificar la orientación cuando la fotografía se encuentra rotada. Para obtener resultados apropiados, se debe considerar que técnicas de fotografía computacional y los procedimientos de reconocimiento incorporan el procesamiento digital de imágenes como parte del desarrollo de la visión por computador (Robles, 2016).

Figura 11

Procesamiento de imágenes



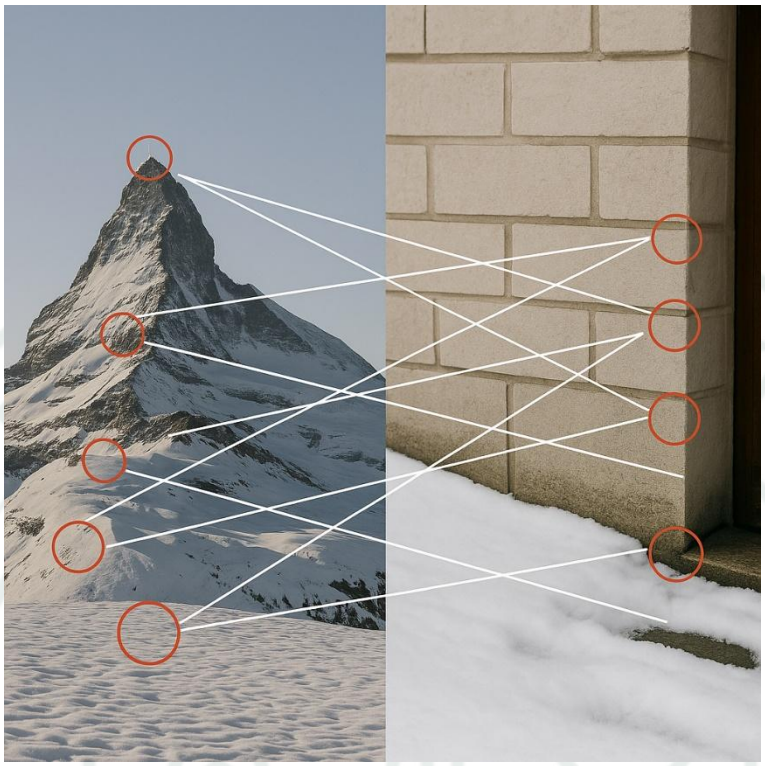
Nota. Elaboración propia.

- **Detección y coincidencia de características**

Se consideran elementos necesarios en diversas aplicaciones de visión por computadora. El primer tipo de característica que puede identificarse corresponde a posiciones concretas dentro de las imágenes, como la cima de una montaña, la esquina de un edificio, el contorno de una puerta o formas llamativas sobre la nieve. A este tipo de característica localizada se le denomina con frecuencia punto clave o punto de interés (incluso esquina), y suele describirse mediante la apariencia de los bloques de píxeles que rodean dicha posición (Szeliski, 2010, p. 207).

Figura 12

Detección y coincidencia de características



Nota. Elaboración propia.

- **Segmentación**

Szeliski (2010) señala que la segmentación de imágenes consiste en identificar conjuntos de píxeles que pertenecen a una misma agrupación. En el campo de la estadística, este problema se conoce como "análisis de clústeres" y ha sido ampliamente investigado, contando con numerosos algoritmos. Del mismo modo, indica que la segmentación de imágenes es uno de los temas más antiguos y estudiados en el área. Las primeras metodologías suelen emplear la división o la unión de regiones, lo que se relaciona con los métodos de partición y agrupamiento mencionados en la literatura de agrupamiento. Técnicas más recientes buscan optimizar criterios globales, como la uniformidad dentro de las regiones y la longitud o diferencia de los bordes entre regiones.

Figura 13

Segmentación de imágenes



Nota. Elaboración propia.

- **Alineación basada en características**

Después de obtener las características de las imágenes, Szeliski (2010) señala que la siguiente fase en diversos algoritmos de visión consiste en hacer corresponder dichas características entre distintas imágenes. Una parte relevante de este proceso consiste en comprobar si el conjunto de características emparejadas es coherente desde el punto de vista geométrico, por ejemplo, si los desplazamientos entre ellas pueden representarse mediante una transformación geométrica 2D o 3D sencilla. Los movimientos obtenidos pueden emplearse luego en otras tareas, como la unión de imágenes o la realidad aumentada. Asimismo, implica el estudio del registro geométrico de imágenes, es decir, el cálculo de transformaciones 2D y 3D que permiten mapear las características de una imagen a otra.

Un caso particular de este problema es la estimación de la postura, que se refiere a determinar la ubicación de una cámara con respecto a un objeto o escena 3D conocida. Otro caso corresponde al cálculo de la calibración interna de una cámara, relacionado con parámetros como la distancia focal y la distorsión radial. Posteriormente, pueden abordarse problemas asociados, como la estimación de la estructura 3D de puntos a partir de coincidencias 2D (triangulación) y la estimación conjunta de la geometría 3D y el movimiento de la cámara (estructura a partir del movimiento) (p. 311).

2.2.2 Billetes falsificados

Un billete falso es aquel que ha sido elaborado sin autorización de la entidad emisora oficial, imitando los elementos físicos de un billete legítimo con la intención de hacerlo pasar como auténtico dentro del sistema económico. Su producción se basa en técnicas de

reproducción gráfica, impresión digital o métodos que buscan replicar colores, tamaños y detalles de seguridad del papel moneda. Estos billetes pueden presentar similitudes visuales considerables respecto a los verdaderos, pero suelen carecer de características avanzadas como relieves calcográficos precisos, microtextos definidos o tintas especiales. La presencia de billetes falsificados representa un riesgo para la circulación monetaria, ya que afecta transacciones comerciales, genera pérdidas económicas y requiere mayores medidas de control por parte de autoridades bancarias y organismos de seguridad (Aguilar y González, 2021).

La identificación de un billete falso se apoya en la observación de elementos de seguridad incorporados en los billetes auténticos. Entre estos se consideran relieves táctiles, microimpresiones, marcas de agua, hilos de seguridad, tintas de variación óptica y patrones geométricos complejos. Los billetes falsificados suelen presentar inconsistencias en contraste, textura del papel o bordes menos definidos, lo que permite diferenciarlos mediante una verificación detallada. Para un análisis básico pueden emplearse procedimientos visuales o lámparas de luz ultravioleta, mientras que evaluaciones más avanzadas recurren a equipos de inspección y sistemas de procesamiento digital de imágenes. El fortalecimiento de estos mecanismos busca evitar que billetes no genuinos ingresen en actividades comerciales, protegiendo a compradores, vendedores y entidades bancarias de pérdidas o reclamos asociados a dinero sin validez legal (Cerezo y Fajardo, 2018).

Figura 14

Billete falso



Nota. Tomado de (Romero, 2025).

Los métodos de detección automática de billetes falsos se desarrollan utilizando algoritmos de visión artificial que analizan imágenes del billete y clasifican su autenticidad. Estas técnicas permiten procesar características como relieve simulado, texturas, bordes y patrones de color. En diversos estudios se emplean filtros, técnicas de segmentación, transformaciones espaciales y clasificadores como Máquinas de Soporte Vectorial (SVM), árboles de decisión o redes neuronales, con resultados que pueden superar el 97% de precisión cuando se usan conjuntos adecuados de entrenamiento. El procedimiento incluye adquisición de la imagen, extracción de características relevantes, comparación con patrones auténticos y decisión automatizada. Su aplicación es útil en cajeros automáticos, máquinas de conteo y establecimientos comerciales, donde se requiere verificar de manera rápida y objetiva la validez del billete recibido (Huaytalla y Humari, 2020).

La circulación de billetes falsos afecta el funcionamiento normal del sistema económico, ya que introduce dinero sin respaldo emitido por la autoridad monetaria. Esto perjudica a comerciantes y ciudadanos que, al recibir billetes no auténticos, asumen pérdidas directas porque estos carecen de valor legal. Además, incrementa los costos de prevención, fiscalización y retiro de billetes fraudulentos. Instituciones como bancos centrales desarrollan campañas de información para que la población conozca los elementos de seguridad del papel moneda, así como programas tecnológicos para la verificación automática. El control y detección oportuna de falsificaciones busca preservar la confianza en la moneda, garantizar transacciones seguras y fortalecer los mecanismos de supervisión financiera, reduciendo riesgos en actividades comerciales y evitando que la falsificación se convierta en una práctica de mayor incidencia (Huaytalla y Humari, 2020).

La circulación de billetes falsos afecta el funcionamiento normal del sistema económico, ya que introduce dinero sin respaldo emitido por la autoridad monetaria. Esto perjudica a comerciantes y ciudadanos que, al recibir billetes no auténticos, asumen pérdidas directas porque estos carecen de valor legal. Además, incrementa los costos de prevención, fiscalización y retiro de billetes fraudulentos. Instituciones como bancos centrales desarrollan campañas de información para que la población conozca los elementos de seguridad del papel moneda, así como programas tecnológicos para la verificación automática. El control y detección oportuna de falsificaciones busca preservar la confianza en la moneda, garantizar transacciones seguras y fortalecer los mecanismos de supervisión financiera, reduciendo riesgos en actividades

comerciales y evitando que la falsificación se convierta en una práctica de mayor incidencia (Huaytalla y Humari, 2020).

La circulación de billetes o monedas falsificadas representa una alteración grave del sistema financiero, ya que introduce medios de pago no autorizados que interfieren en el normal desarrollo de las transacciones económicas. En estos casos, el dinero adulterado suele presentar imitaciones deficientes de las medidas de seguridad oficiales, lo que, al pasar inadvertido, ocasiona pérdidas directas para comerciantes y consumidores. Además, esta actividad ilícita alimenta redes delictivas que operan de manera clandestina y afectan la credibilidad del medio circulante. Por ello, el fortalecimiento constante de los elementos de seguridad en la fabricación de billetes y monedas, junto con campañas de información ciudadana, se vuelve indispensable para detectar, reducir y prevenir la aceptación de dinero fraudulento en el mercado (Barrionuevo, 2024).

2.2.3 Características visuales de los billetes

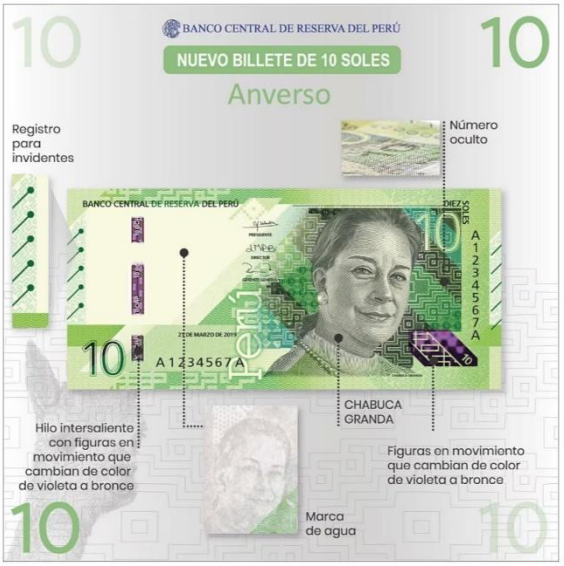
Los billetes peruanos incluyen zonas impresas con una tinta especial que modifica su tonalidad cuando el papel se inclina bajo distintas fuentes de luz. Este elemento funciona como un mecanismo visual de verificación inmediata para el usuario, ya que permite identificar variaciones perceptibles que un falsificador difícilmente reproduce con precisión. En el documento se muestran ejemplos de esta característica en billetes de 50, 100 y 200 soles, donde se aprecia el cambio de tono al girar el billete. Esta tinta forma parte de las medidas de seguridad de fácil revisión que refuerzan la autenticidad del papel moneda y complementan otros elementos como microimpresiones, registro perfecto y marca de agua (Aguilar, 2016).

El uso de tinta que cambia de color no se aplica de forma uniforme, sino en áreas puntuales del diseño del billete, normalmente asociadas a denominaciones o figuras geométricas. Esto permite al usuario asociar un patrón visual cuando inclina el billete y confirmar si el cambio de color corresponde al comportamiento esperado. La ubicación estratégica ayuda a que el público pueda validar rápidamente la autenticidad sin herramientas externas. Esta característica es aplicada de manera diferenciada según la denominación, reforzando su función como marcador visual de seguridad (Huaytalla y Humari, 2020).

La tinta que cambia de color está diseñada para resistir intentos de copia o simulación mediante impresoras convencionales, fotocopiadoras o métodos de impresión digital utilizados en falsificaciones. Este tipo de tinta requiere procesos industriales y materiales especializados que no están disponibles para el público común, lo que dificulta su imitación. Según el documento, es uno de los elementos clave que permiten diferenciar billetes auténticos de reproducciones ilícitas al observar discrepancias de tono, brillo o ausencia total de variación cromática en billetes falsos (Ordoñez, 2017).

Figura 15

Elementos de seguridad del billete de 10 soles



Nota. Tomado de (Granda, 2021).

La figura 15 muestra el anverso del nuevo billete peruano de 10 soles emitido por el Banco Central de Reserva del Perú e identifica sus principales elementos de seguridad visibles utilizados para comprobar su autenticidad y dificultar su falsificación. Se aprecia el registro para invidentes, que consiste en impresiones en relieve perceptibles al tacto para facilitar el reconocimiento del billete. También se observa el número oculto, visible únicamente al inclinar el billete, así como un hilo intersaliente de seguridad con figuras en movimiento que cambian de color de violeta a bronce al girarlo. De igual manera, se señalan otras figuras con cambio de color mediante tintas ópticamente variables y la marca de agua visible al trasluz, donde se distingue el retrato de la cantante peruana Chabuca Granda, personaje principal representado en esta denominación. Estos elementos trabajan de forma conjunta para ofrecer mayores

garantías de autenticidad y permitir una verificación rápida por parte de los usuarios y del sistema financiero.

Figura 16

Elementos de seguridad del billete de 20 soles



Nota. Tomado de (Granda, 2021).

La figura 16 presenta el diseño del nuevo billete peruano de 20 soles, parte de la reciente familia de billetes emitida por el Banco Central de Reserva del Perú, y muestra sus principales elementos de seguridad tanto en el anverso como en el reverso. En el anverso se observan características como el registro para invidentes, conformado por relieves táctiles ubicados en el borde del billete, y el número oculto, visible únicamente al inclinar el billete en determinadas posiciones. También se identifica el hilo de seguridad intersaliente con figuras en movimiento que cambian de color de verde a azul, además de las tintas ópticamente variables que modifican su tonalidad al girar el billete. Se distingue igualmente la marca de agua con el retrato del escritor peruano José María Arguedas y el denominativo “20” con efecto de cambio de color. En el reverso se muestra la imagen de un cóndor, ave emblemática de los Andes, junto con el escudo de armas y otros elementos gráficos que refuerzan la identidad cultural del billete. En conjunto, la figura destaca las innovaciones en seguridad y diseño destinadas a facilitar la verificación del billete, prevenir falsificaciones y promover símbolos representativos del patrimonio peruano.

Figura 17

Elementos de seguridad del billete de 50 soles



Nota. Tomado de (Granda, 2021).

La figura 17 muestra el diseño del nuevo billete peruano de 50 soles, perteneciente a la nueva familia de billetes emitidos por el Banco Central de Reserva del Perú, destacando los principales elementos de seguridad presentes en su anverso y reverso. En el anverso se observan mecanismos de verificación como el registro para invidentes, el cual consta de relieves táctiles en el borde del billete, así como el número oculto, visible solo al inclinar el billete en determinados ángulos. También se aprecia el hilo de seguridad intersaliente con figuras en movimiento y una tonalidad marrón brillante, además de elementos con cambio de color que varían de dorado a verde al girar el billete. En la zona central se identifica la marca de agua con el retrato de la destacada historiadora María Rostworowski, junto con el escudo de armas del Perú y la denominación “50” impresa con efecto de cambio de color. En el reverso se muestra la ilustración de un jaguar, especie representativa de la fauna nacional, acompañada de detalles y tramas de seguridad que refuerzan la identidad visual del billete y dificultan su falsificación. En conjunto, la figura presenta un diseño moderno y seguro, orientado a facilitar la autenticación del billete y resaltar elementos históricos y naturales del país.

Figura 18

Elementos de seguridad del billete de 100 soles



Nota. Tomado de (Granda, 2021).

La figura 18 presenta el diseño del nuevo billete peruano de 100 soles en su anverso, emitido por el Banco Central de Reserva del Perú, destacando sus principales elementos de seguridad. En el centro del billete se encuentra el retrato de Pedro Paulet, pionero de la aeronáutica, acompañado de la denominación en números y texto. Entre los dispositivos de seguridad visibles se incluye el registro para invidentes mediante relieves táctiles, así como un número oculto que puede observarse al inclinar el billete. También se aprecia un hilo de seguridad intersaliente con figuras en movimiento que cambian de color de fucsia a verde, además de elementos adicionales con el mismo efecto de movimiento y variación cromática. En la zona izquierda se observa la marca de agua con el rostro del personaje, perceptible a contraluz. Todos estos componentes permiten verificar la autenticidad del billete y dificultar su falsificación, combinando características visuales, táctiles y ópticas que refuerzan la seguridad del nuevo diseño monetario peruano.

2.2.3.1 Arquitectura de la red neuronal artificial convolucional

La arquitectura de una red neuronal se entiende como la forma en que está organizada una red neuronal convolucional. Según Zarate (2022), su estructura se compone de una capa de entrada, una capa de salida y varias capas ocultas intermedias. Estas capas aplican procesos que modifican la información con el propósito de aprender rasgos específicos de los datos. Las tres capas que se utilizan con mayor frecuencia son: convolución, activación o ReLU y pooling.

- Convolución: Aplica un conjunto de filtros sobre las imágenes de entrada, activando determinados rasgos presentes en ellas.
- ReLU o unidad lineal rectificada: Facilita un entrenamiento más eficiente asignando los valores negativos a cero y manteniendo los positivos. Se le denomina activación porque solo las características activadas continúan hacia la siguiente capa.
- Pooling: Reduce la información resultante disminuyendo la densidad de muestreo de manera no lineal, lo que permite disminuir la cantidad de parámetros que la red debe aprender.

Estas operaciones se ejecutan en múltiples capas, de manera que cada una de ellas va aprendiendo a reconocer distintos atributos dentro de los datos procesados

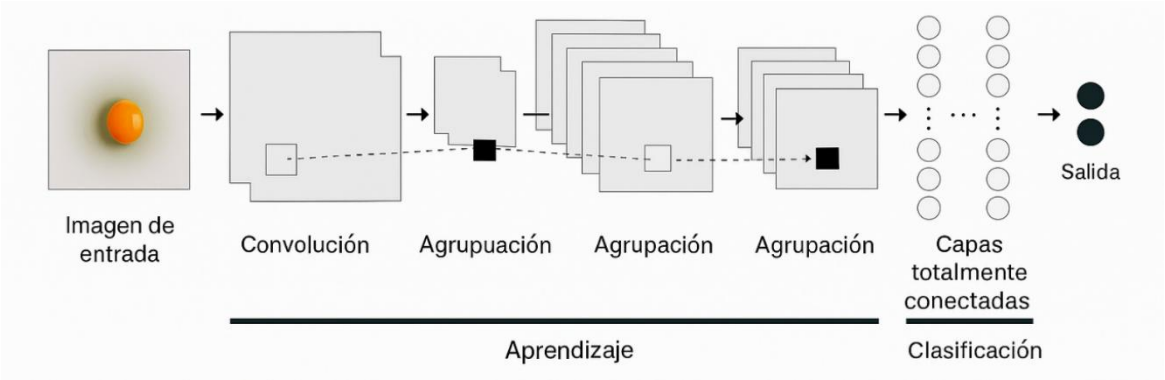
2.2.4 Convolutional Neural Network (Red Neuronal Convolucional)

Una CNN es un método de aprendizaje profundo que simula el funcionamiento del cerebro humano para formar patrones y analizar datos con el fin de apoyar procesos de decisión. La estructura de una CNN permite aplicar operaciones de convolución mediante capas de distintos tamaños, lo que facilita la captura de una variedad amplia de rasgos, desde aspectos generales de la imagen hasta detalles más específicos (figura 19). Esto hace posible clasificar cultivos y reconocer enfermedades agrícolas al analizar la imagen completa (Oscco-Ccuiro et al., 2024).

Desde su aparición en la década de 1990, se han desarrollado diversos modelos de CNN, entre ellos AlexNet, VGG Net, ZF Net, GoogLeNet y las redes totalmente convolucionales (FCN). Estos modelos han mostrado resultados relevantes en múltiples áreas, destacando su alto rendimiento en la detección de enfermedades en cultivos agrícolas (Oscco-Ccuiro et al., 2024).

Figura 19

Arquitectura general de una red neuronal convolucional (CNN)



Nota. Elaboración propia.

Se trata de una tecnología reciente que ha mostrado un desempeño adecuado en tareas de reconocimiento cuando se trabaja con grandes volúmenes de datos, evitando la necesidad de elaborar manualmente funciones específicas. Al emplear una red neuronal convolucional, una imagen se interpreta como una matriz tridimensional, donde la dimensión de profundidad corresponde a los canales de color rojo, verde y azul, excepto si es una imagen en escala de grises, la cual debe transformarse a una matriz 3D. Diversas estructuras de redes convolucionales utilizan conversiones de color como RGB, HSV y HSB. En la Figura 9 se observan distintos tipos de filtros utilizados en este proceso (Cifuentes et al., 2019).

Tabla 2

Etapas del procesamiento en una Red Neuronal Convolucional (CNN)

Etapa del proceso	Descripción breve	Resultado esperado
Entrada	Se proporciona una imagen en formato digital que contiene los objetos o patrones a analizar.	Datos visuales listos para ser procesados.
Convolución	Se aplican filtros (kernels) que recorren la imagen para detectar características locales como bordes, colores o texturas.	Primeros mapas de características relevantes.
Agrupación (Pooling)	Se reduce el tamaño de los mapas de características, conservando la información más representativa y disminuyendo el costo computacional.	Mapas reducidos y más comprensibles para la red.
Convolución y Agrupación adicional	Se repiten estas operaciones varias veces para extraer características más profundas y complejas.	Mapas de características de mayor nivel semántico.

Capas totalmente conectadas (Clasificación)	Las características extraídas se transforman en vectores que permiten clasificar o reconocer patrones.	Probabilidades asociadas a cada categoría.
Salida	La red asigna una clase o decisión final basada en el análisis anterior.	Etiqueta o respuesta del modelo (predicción).

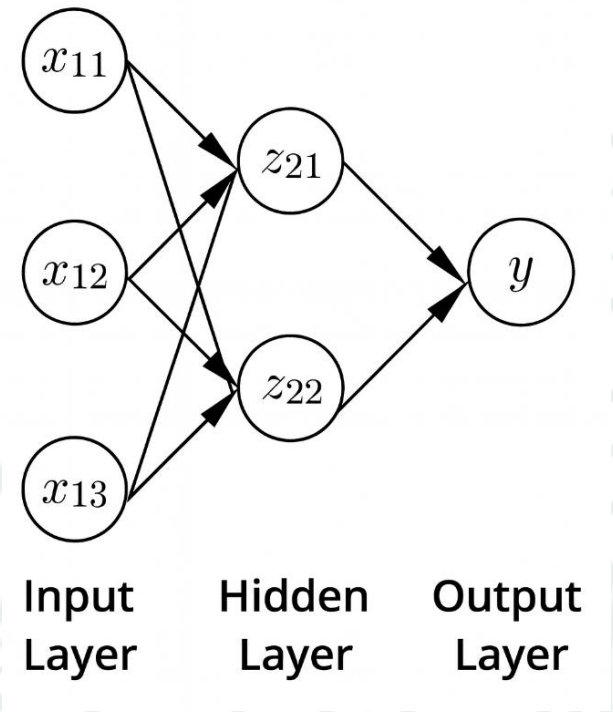
Nota. Tomado de (Diaz y Vilcas, 2022).

Antes de explicar las redes neuronales convolucionales, resulta útil revisar primero el modelo básico conocido como red neuronal alimentada hacia adelante. En un contexto de aprendizaje supervisado, se dispone de un conjunto de datos etiquetados (x_i, y_j), donde x_i representa las características y y_j las etiquetas correspondientes para cada ejemplo de entrenamiento. En términos generales, las redes neuronales proporcionan una forma avanzada de representación mediante una función no lineal $h_W(x)$ aplicada a la variable de entrada x. Esta función depende de una matriz de pesos W, cuyos valores se determinan para obtener un ajuste adecuado a los datos. En la figura 3.2 se observa una red neuronal simple conformada por tres neuronas de entrada x_{11}, x_{12}, x_{13} y una salida definida por $y = h_{W(x_{11},x_{12},x_{13})}$ (Cifuentes et al., 2019).

Además, es importante señalar que este tipo de red busca aproximar relaciones entre los datos de entrada y las respuestas esperadas mediante un proceso de ajuste progresivo de los pesos. Durante el entrenamiento, el modelo compara sus predicciones con los valores reales y calcula un error, el cual se utiliza para modificar los parámetros internos y mejorar su precisión en iteraciones posteriores. Este procedimiento se conoce como retropropagación del error y constituye uno de los mecanismos centrales en el aprendizaje de redes neuronales. De esta manera, el modelo no solo almacena información, sino que aprende patrones subyacentes presentes en los datos, permitiendo generalizar su comportamiento ante nuevos ejemplos que no formaron parte del conjunto inicial de entrenamiento (Zarate, 2022).

Figura 20

Red neuronal pre-alimentada



Nota. Elaboración propia.

2.2.5 Recurrente Neural Network (Red Neuronal Recurrente)

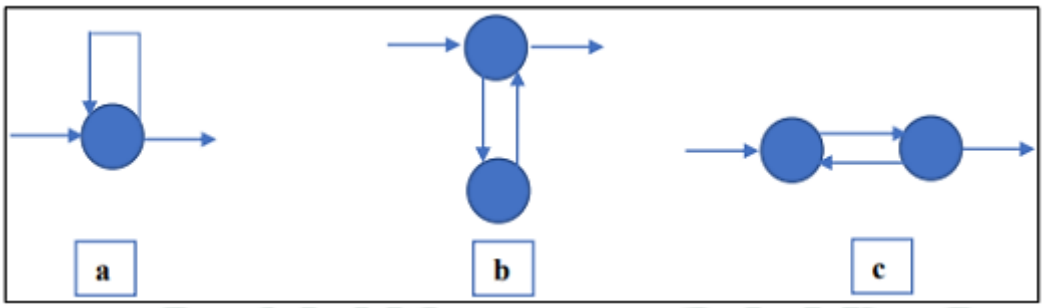
Una Red Neuronal Recurrente (Recurrent Neural Network - RNN) es un modelo de red neuronal orientado a trabajar con secuencias de datos, en el cual la salida generada en un momento determinado depende tanto de la entrada presente como de las entradas previas que forman parte de la misma secuencia (Cifuentes et al., 2019).

De acuerdo con Arana (2021) este tipo de red se distingue por generar ciclos dentro de sus neuronas a través de conexiones recurrentes, lo que la diferencia del perceptrón. Estas conexiones en forma de bucle pueden presentarse de las siguientes maneras:

- La neurona se conecta consigo misma.
- Existen conexiones entre las neuronas que pertenecen a una misma capa.
- Se establecen conexiones desde las neuronas de una capa hacia la capa previa.

Figura 21

Esquema de conexiones recurrentes en una RNN



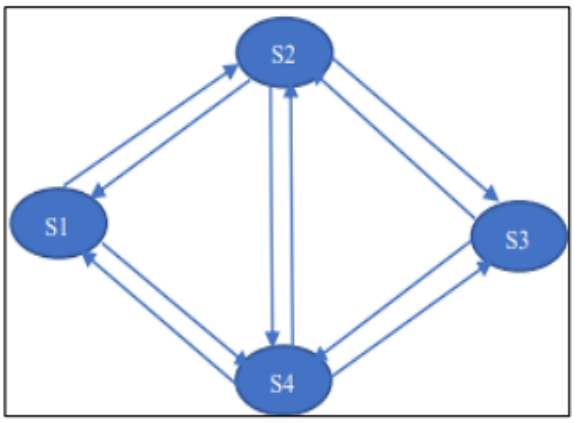
Nota. Tomado de (Chafla, 2019).

Al incorporar conexiones recurrentes (ciclos), la activación de una neurona deja de depender únicamente de las neuronas de la capa previa, como ocurre en el perceptrón multicapa, y pasa también a relacionarse con el estado de activación de otras neuronas de la misma red a las que esté conectada (Arana, 2021). Se reconocen tres clases de redes recurrentes, las cuales son:

- **Red Hopfield:** Propuesta por John Hopfield en la década de 1980, se considera un modelo de memoria asociativa de patrones o muestras, ya que puede reconstruir información almacenada incluso cuando esta se presenta de manera incompleta o con presencia de ruido. Su principal característica es que cada neurona se conecta con todas las demás, con la excepción de sí misma (Arana, 2021).

Figura 22

Estructura de conexiones en una Red Hopfield



Nota. Tomado de (Chafla, 2019).

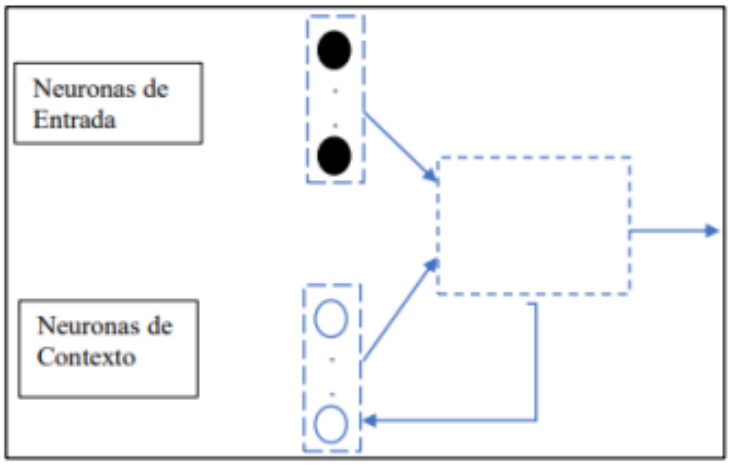
- **Red parcialmente recurrente**

Se trata de redes con múltiples capas que se distinguen porque cuentan con dos tipos de entradas hacia sus neuronas. La primera corresponde a las neuronas de entrada, encargadas de recibir la información proveniente del entorno. La segunda está formada por las neuronas de contexto, cuya función es almacenar datos obtenidos de las conexiones recurrentes, operando como una forma de memoria dentro de la red. En las redes parcialmente recurrentes, dichas conexiones recurrentes suelen establecerse de manera uno a uno, es decir, desde una neurona de la red hacia una sola neurona de contexto (Cabezón, 2018).

Un concepto complementario indica que este tipo de redes utiliza una estructura donde parte de la información procesada se conserva temporalmente para ser reutilizada en pasos posteriores. Este mecanismo permite que el modelo no solo analice los datos actuales, sino también patrones previos, facilitando el reconocimiento de secuencias o eventos que dependen del tiempo. En este enfoque, las neuronas adicionales almacenan estados anteriores y los integran al cálculo de nuevas salidas, lo cual mejora la capacidad de predicción en procesos donde existe continuidad o relación entre entradas consecutivas. Así, estas redes logran un comportamiento más cercano al procesamiento secuencial observado en diversos sistemas de señal y aprendizaje automático (Cabezón, 2018).

Figura 23

Diagrama de red parcialmente recurrente



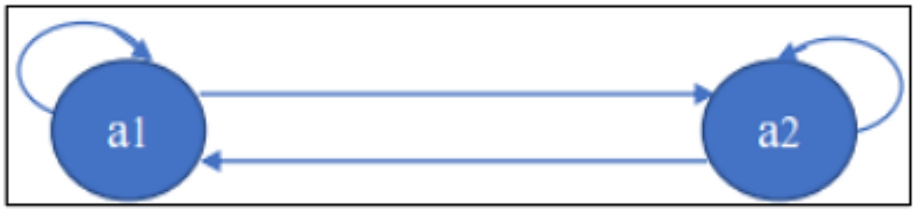
Nota. Tomado de (Chafla, 2019).

- **Red totalmente recurrente**

Las redes completamente recurrentes se distinguen porque las neuronas que las conforman toman como entrada tanto la activación proveniente de otras neuronas de la misma red como su propia activación previa. Este tipo de configuración permite que la información circule internamente, lo que hace posible que la red mantenga datos de estados anteriores y procese relaciones temporales entre elementos de una secuencia. De esta manera, las salidas no dependen únicamente de la entrada actual, sino también de la memoria generada en pasos anteriores (Cabezón, 2018). El modelo general de esta arquitectura se representa del siguiente modo:

Figura 24

Esquema de una red totalmente recurrente



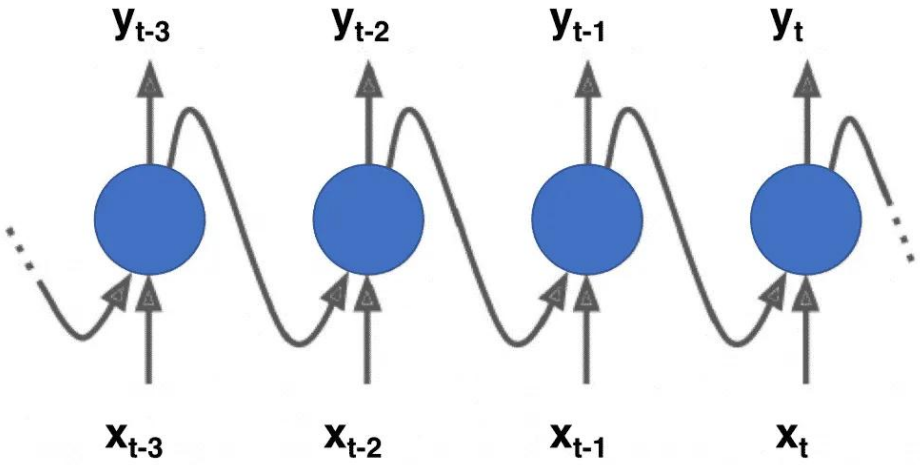
Nota. Tomado de (Chafla, 2019).

2.2.5.1 Tipos:

a) **Redes Neuronales Recurrentes Básicas:** Estas redes utilizan estructuras recurrentes básicas, como las neuronas con capacidad de mantener información temporal breve, conocidas como LSTM, o las unidades con mecanismos de puertas internas, denominadas GRU. Su finalidad es reconocer dependencias que se extienden a lo largo de una secuencia de datos, permitiendo analizar cómo un elemento se relaciona con otros situados antes o después dentro del mismo conjunto. Gracias a este tipo de unidades es posible procesar información secuencial conservando aspectos relevantes del contexto, lo que facilita el tratamiento de tareas en las que es importante comprender patrones que no se limitan a posiciones inmediatas (Cabezón, 2018).

Figura 25

Redes Neuronales Recurrentes Básicas

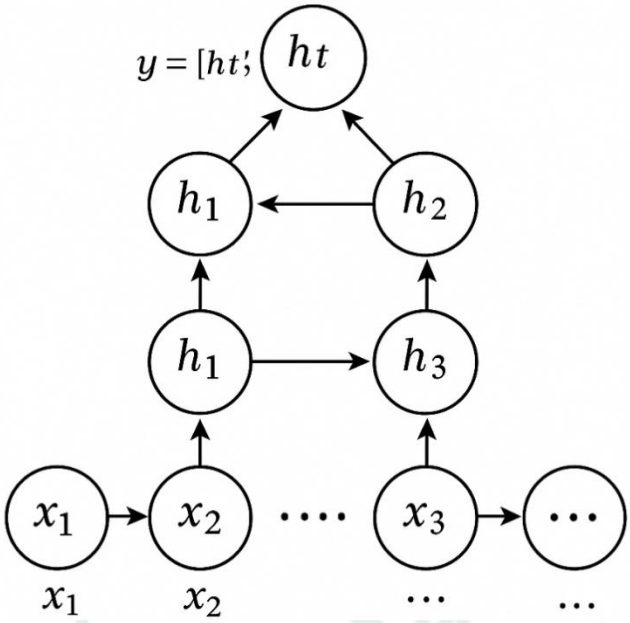


Nota. Elaboración propia.

b) Redes Neuronales Recurrentes Bidireccionales: Las redes bidireccionales integran conexiones que hacen posible que la información sea analizada en dos direcciones dentro de una misma secuencia, es decir, desde el inicio hacia el final y de forma inversa. Este mecanismo proporciona un contexto más amplio al modelo, ya que no solo considera los datos anteriores, sino también los que se encuentran después del punto evaluado. Gracias a esta capacidad, se mejora la precisión en tareas de predicción, clasificación o interpretación de patrones, puesto que el sistema puede apoyarse en una visión más completa de la información, lo que favorece la toma de decisiones en función del contenido total disponible (Cabezón, 2018).

Figura 26

Redes Neuronales Recurrentes Bidireccionales



Nota. Elaboración propia.

La figura representa el funcionamiento de una Red Neuronal Recurrente Bidireccional (Bi-RNN), donde la información fluye en dos direcciones: hacia adelante y hacia atrás, permitiendo que el modelo tenga en cuenta tanto el contexto previo como el posterior de una secuencia.

En la parte inferior se observan las entradas $x_1, x_2, x_3, ...$, que representan los elementos de una secuencia (por ejemplo, palabras en una oración o valores temporales en una serie). Cada entrada es procesada por dos neuronas recurrentes: una que avanza en la secuencia y otra que retrocede. Estas generan estados ocultos $h_1, h_2, h_3, ...$, que se conectan entre sí para mantener información del pasado y del futuro.

En el nivel superior, los estados ocultos provenientes de ambas direcciones se combinan para producir el estado final h_t, que a su vez se utiliza para generar la salida $y = [h_t]$. Esto permite que el modelo tome decisiones basadas en mayor contexto que una RNN tradicional, lo que resulta especialmente útil en tareas como reconocimiento de voz, traducción automática y análisis de secuencias donde el significado depende de elementos anteriores y posteriores.

CAPÍTULO III

PLANTEAMIENTO DE LA INVESTIGACIÓN

3.1 Metodología de Investigación

La investigación emplea un enfoque cuantitativo, basado en el análisis numérico del desempeño de modelos de aprendizaje profundo aplicados a la identificación de billetes peruanos falsificados. Se trabaja mediante la recolección y procesamiento de imágenes, el entrenamiento de modelos CNN y RNN por separado y en forma híbrida, y la evaluación mediante métricas de clasificación. El estudio sigue una secuencia estructurada: recopilación del dataset, preprocesamiento de imágenes, definición de arquitecturas, entrenamiento, validación y comparación de resultados.

3.2 Tipo de Investigación

La presente investigación es de tipo aplicada, porque emplea técnicas de visión por computadora, procesamiento digital de imágenes y aprendizaje profundo para abordar un problema concreto relacionado con el reconocimiento automático de billetes peruanos. Del mismo modo, tiene un carácter exploratorio, ya que examina el comportamiento de distintas arquitecturas de aprendizaje profundo, como CNN, CNN-LSTM y CNN-GRU, con la finalidad de identificar cuál ofrece mejores resultados en el conjunto de datos analizado. En ese sentido, el estudio no busca establecer relaciones causales definitivas, sino generar una base experimental inicial y aportar evidencia sobre la utilidad de estas técnicas en la clasificación de billetes peruanos.

3.3 Diseño de Investigación

Se utiliza un diseño experimental, pues se manipulan variables relacionadas con la arquitectura del modelo (CNN, RNN y CNN-RNN) y se observan sus efectos sobre la capacidad de identificación de billetes falsificados. Se desarrollan distintos escenarios de prueba según iluminación, ángulos y desgaste del billete para evaluar el rendimiento del modelo. El diseño considera fases de entrenamiento, validación y prueba usando un conjunto de datos dividido en proporciones establecidas (train/validation/test).

3.4 Técnicas e Instrumentos

3.4.1 Técnicas

Para el desarrollo de la investigación se emplearán las siguientes técnicas:

a) **Revisión documental:** Para recopilar información secundaria proveniente de artículos científicos, reportes del BCRP, normas y estudios previos relacionados con falsificación de billetes y modelos de aprendizaje profundo.
b) **Análisis de contenido:** Para examinar y clasificar la información obtenida de fuentes documentales y determinar los criterios visuales relevantes de los billetes peruanos.
c) **Análisis cuantitativo:** Para procesar los datos numéricos derivados del entrenamiento y evaluación de los modelos CNN, RNN y CNN-RNN.
d) **Experimentación computacional:** Para ejecutar las pruebas de entrenamiento, evaluación y comparación de los modelos en el entorno de programación.

3.4.2 Instrumentos

Los instrumentos que se emplearán son:

- **Dataset de imágenes de billetes peruanos:** Conjunto estructurado de imágenes usadas como fuente de datos secundaria.
- **Visual Studio Code:** Entorno de desarrollo para la construcción y ejecución de los modelos.
- **Python:** Lenguaje de programación principal para el procesamiento de datos y la implementación de las redes neuronales.

3.5 Campo de Verificación

3.5.1 Ubicación Temporal

La investigación se llevará a cabo durante el año 2025 - 2026.

3.5.2 Ubicación espacial

El estudio se desarrollará en un entorno personal, utilizando equipos ubicados en el lugar de trabajo del investigador.

3.5.3 Unidades de Estudio

- Imágenes de billetes peruanos auténticos en diferentes condiciones de iluminación, desgaste y orientación.

- Imágenes de billetes peruanos falsificados, clasificadas como tales en el dataset utilizado.
- Características visuales extraídas por los modelos, como patrones, texturas, regiones y secuencias procesadas por la CNN y la RNN.

CAPÍTULO IV

DISEÑO E IMPLEMENTACIÓN DE LA ARQUITECTURA HÍBRIDA DE APRENDIZAJE PROFUNDO

4.1. Dataset de billetes peruanos

El desarrollo del sistema de reconocimiento de billetes peruanos requirió la utilización de un conjunto de datos estructurado y previamente organizado por clases. Para ello, se empleó el repositorio Dataset-Billetes-peruanos, descargado desde GitHub y almacenado localmente en el entorno de trabajo. Este conjunto de datos contiene imágenes distribuidas en carpetas correspondientes a las fases de entrenamiento, validación y prueba, siguiendo una organización común en problemas de clasificación de imágenes.

La estructura del dataset se compone de tres directorios principales: training_set, validation_set y testing_set. A partir del contenido de la carpeta de entrenamiento se identificaron las categorías disponibles, obteniéndose un total de 16 clases. En términos generales, cada clase representa una categoría visual distinta dentro del problema de reconocimiento planteado. La determinación del número de clases se realizó mediante la expresión:

$$C = \text{len}(\text{Categorías})$$

donde C representa el número total de clases contenidas en el conjunto de datos. En este caso:

$$C = 16$$

Posteriormente, se efectuó el conteo de imágenes por partición y por categoría. Para ello, el algoritmo recorrió cada carpeta y contabilizó los archivos con extensiones de imagen válidas, tales como .jpg, .jpeg y .png. Matemáticamente, el número de imágenes por subconjunto puede expresarse como:

$$N_s = \sum_{c=1}^{C} n_{c,s}$$

donde:

- N_s: número total de imágenes en el subconjunto s,
- $n_{c,s}$: número de imágenes de la clase c en el subconjunto s,
- C: número total de clases.

A partir del análisis realizado, se obtuvieron las siguientes estadísticas generales del dataset:

- Número de categorías: 16
- Imágenes de entrenamiento: 6,568
- Imágenes de validación: 2,486

- Imágenes de prueba: 0
- Total de imágenes: 9,054

El número total de imágenes del conjunto se calculó mediante:

$$N_{total} = N_{train} + N_{val} + N_{test}$$

Reemplazando con los valores obtenidos:

$$N_{total} = 6568 + 2486 + 0 = 9054$$

Con la finalidad de interpretar mejor la distribución de datos, también es posible expresar la proporción porcentual de cada subconjunto respecto del total:

$$P_s = \frac{N_s}{N_{total}} \times 100$$

De este modo, para el presente dataset se tiene:

$$P_{train} = \frac{6568}{9054} \times 100 \approx 72.54\%$$

$$P_{val} = \frac{2486}{9054} \times 100 \approx 27.46\%$$

$$P_{test} = \frac{0}{9054} \times 100 = 0\%$$

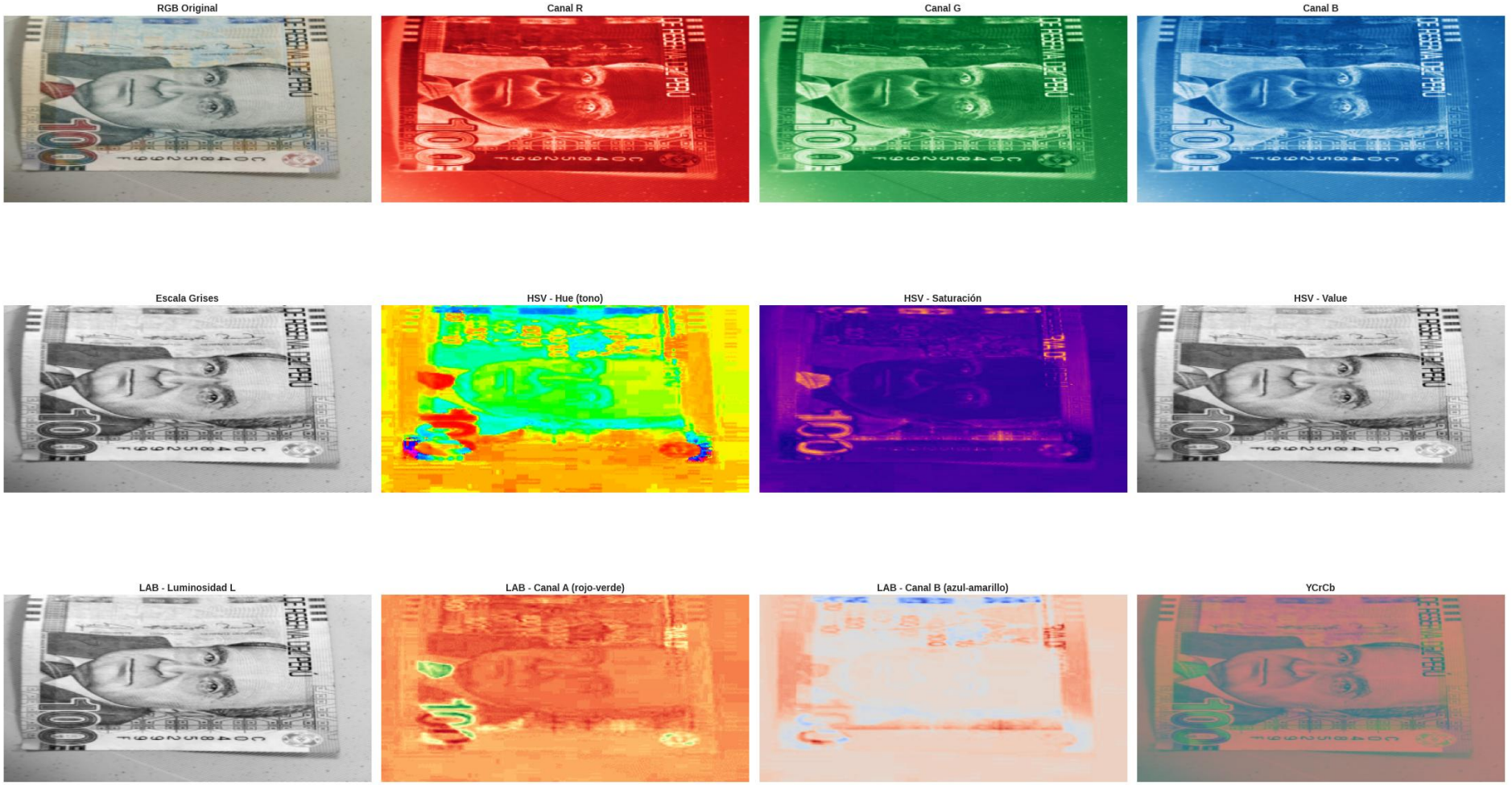
4.2. Características visuales de los billetes peruanos

El análisis de las características visuales de los billetes peruanos constituye una etapa previa al proceso de identificación automática de falsificaciones, debido a que permite reconocer los elementos gráficos y de seguridad que diferencian a un billete auténtico de uno adulterado. Entre estas características se consideran el color, la textura aparente, los bordes, los patrones geométricos, las microimpresiones, la nitidez de los detalles y la distribución de los elementos de seguridad presentes en cada denominación. En el contexto de esta investigación, dichas propiedades visuales representan la información de entrada que será procesada por el modelo de visión por computadora, permitiendo que la arquitectura híbrida CNN-RNN extraiga patrones espaciales y secuenciales asociados a la autenticidad del billete.

La figura 27 presenta la misma imagen de un billete peruano transformada en distintos espacios de color y canales de representación. El propósito de esta comparación es observar cómo cada transformación resalta determinadas propiedades visuales del billete, tales como intensidad, contraste, tonalidad, saturación y composición cromática.

Figura 27

Análisis de espacios de color para clasificación de billetes



Nota. Elaboración propia

A continuación, se explica qué representación ofrece mayor información discriminante para el proceso de clasificación automática entre billetes auténticos y falsificados.

- En la primera fila, la imagen denominada RGB Original corresponde a la fotografía capturada en su forma natural, es decir, en el espacio de color RGB. Esta representación conserva la apariencia visual tal como sería percibida por el ojo humano y sirve como referencia base para el análisis posterior. A partir de ella se descomponen los tres canales principales: Canal R, Canal G y Canal B. El Canal R representa la intensidad del color rojo en cada píxel, permitiendo observar qué regiones del billete contienen mayor contribución de este componente cromático. El Canal G muestra la intensidad del color verde, resaltando zonas donde predominan tonalidades intermedias y detalles gráficos asociados a este canal. El Canal B representa la intensidad del color azul, lo cual resulta útil para identificar diferencias en tintas, fondos y patrones impresos que pueden no ser tan evidentes en la imagen original. En conjunto, estos tres canales permiten analizar por separado la respuesta cromática del billete, lo que favorece la detección de alteraciones de color producidas por falsificaciones o impresiones de baja calidad.

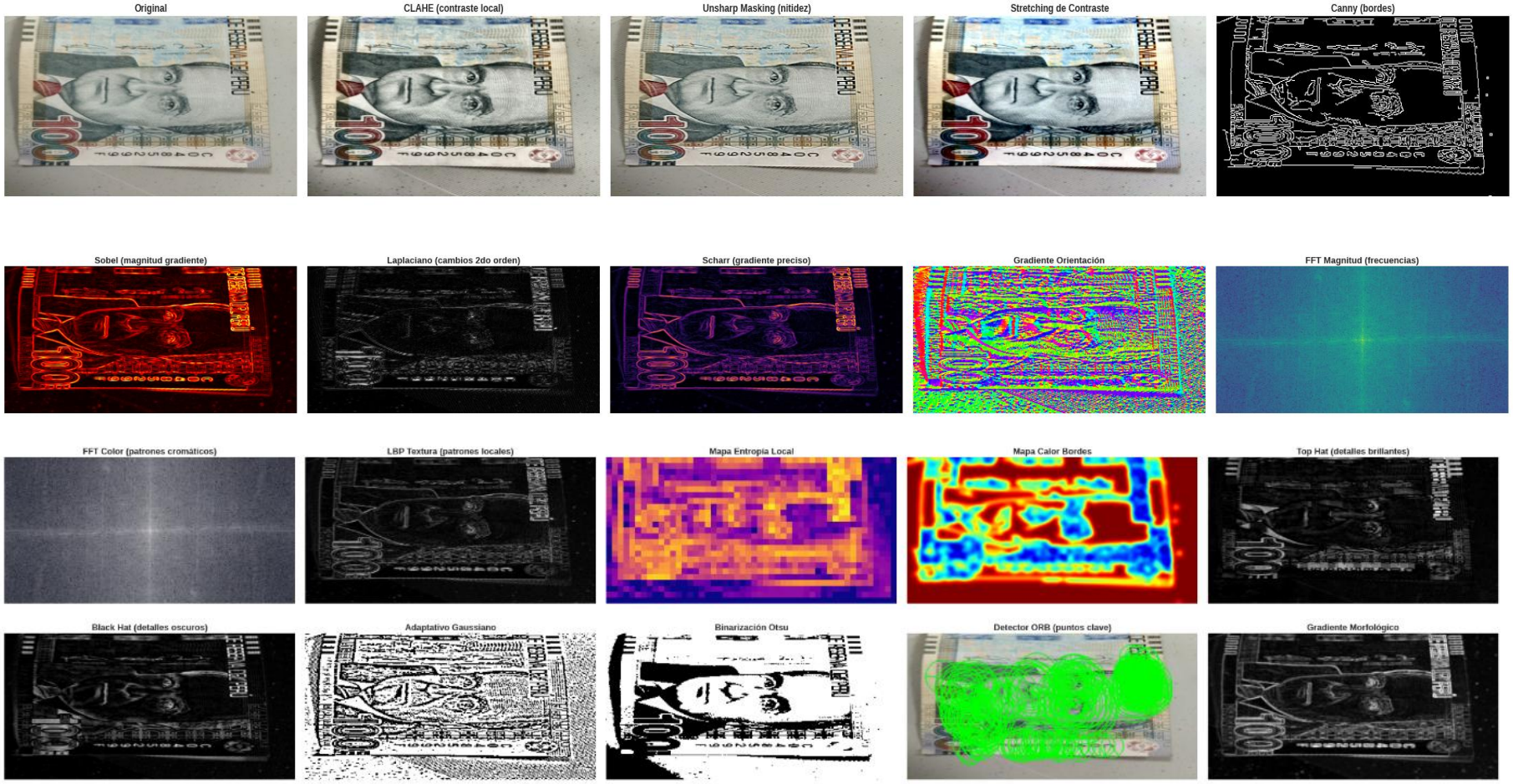
- En la segunda fila, la Escala de Grises representa la imagen utilizando únicamente niveles de intensidad luminosa, eliminando la información de color. Esta conversión simplifica el análisis y permite centrar la atención en bordes, texturas, relieves aparentes, contraste y distribución tonal. A su lado se presenta el espacio HSV, descompuesto en tres componentes. El canal HSV - Hue (tono) representa el tipo de color dominante en cada región, es decir, diferencia si un área tiende hacia rojo, verde, azul u otras tonalidades. Esta componente es útil cuando se desea analizar la composición cromática sin que influya directamente el brillo. El canal HSV - Saturación muestra la pureza o intensidad del color; valores altos indican colores más vivos, mientras que valores bajos corresponden a tonos más apagados o cercanos al gris. Este canal puede ayudar a detectar zonas impresas con distinta calidad o con pigmentos no equivalentes a los del billete original. Finalmente, HSV - Value representa el brillo o nivel de iluminación de la imagen. Esta componente resalta las zonas más claras y oscuras del billete, lo que facilita la observación de contrastes, sombras y reflejos sobre la superficie.

- En la tercera fila se muestran representaciones en los espacios LAB y YCrCb, que son ampliamente utilizados en procesamiento de imágenes por su capacidad para separar luminancia y crominancia. El canal LAB - Luminosidad L describe la claridad de la imagen, desde tonos oscuros hasta tonos claros, sin depender directamente del color. Esta representación es adecuada para estudiar contraste, contornos y visibilidad de detalles impresos. El canal LAB - Canal A (rojo-verde) expresa la variación cromática en el eje que va de verde a rojo. En esta imagen, las regiones con distinta composición cromática respecto a ese eje aparecen realzadas, lo que permite distinguir detalles gráficos que podrían ser relevantes para la autenticación. El LAB - Canal B (azul-amarillo) representa la variación en el eje azul–amarillo, destacando zonas donde esos componentes tienen mayor presencia relativa. Esta separación puede ser útil para examinar diferencias de tinta o patrones cromáticos característicos del billete. Finalmente, la representación YCrCb separa la imagen en luminancia y componentes de crominancia, lo cual facilita el análisis independiente del brillo y del color. Este espacio suele emplearse cuando se requiere robustez frente a cambios de iluminación, ya que permite trabajar con información visual más estable para el entrenamiento de modelos de clasificación.

La figura 28 presenta un conjunto de transformaciones y descriptores aplicados sobre la imagen de un billete peruano con el propósito de resaltar propiedades visuales útiles para la detección de falsificaciones. Cada representación enfatiza un aspecto distinto de la imagen, como contraste, bordes, textura, frecuencia, entropía o puntos clave. En conjunto, estas características permiten analizar con mayor detalle variaciones de impresión, nitidez, distribución de tinta, relieves aparentes, patrones locales y estructuras geométricas que pueden diferenciar un billete auténtico de uno falsificado.

Figura 28

Características para la detección de falsificaciones



Nota. Elaboración propia

- En la primera fila, la imagen Original corresponde al billete capturado sin transformaciones, por lo que representa la referencia base para todas las comparaciones. La imagen CLAHE (contraste local) aplica una ecualización adaptativa del histograma, lo que mejora el contraste en regiones pequeñas de la imagen y permite observar con mayor claridad detalles impresos que podrían pasar desapercibidos en zonas de baja iluminación. La representación Unsharp Masking (nitidez) incrementa la definición de bordes y contornos, resaltando líneas finas, textos y formas de seguridad. La imagen Stretching de Contraste expande el rango tonal, haciendo más visibles las diferencias entre áreas claras y oscuras. Finalmente, Canny (bordes) muestra un mapa binario de bordes detectados, donde se identifican contornos estructurales del rostro, números, textos y marcos del billete.

- En la segunda fila, Sobel (magnitud gradiente) representa la intensidad de cambio en la imagen, es decir, qué tan bruscas son las variaciones entre píxeles vecinos. Esta transformación permite localizar zonas con bordes y cambios de intensidad. La imagen Laplaciano (cambios 2do orden) detecta variaciones más abruptas de intensidad mediante derivadas de segundo orden, siendo útil para resaltar contornos finos y detalles de impresión. La representación Scharr (gradiente preciso) cumple una función similar a Sobel, pero con una estimación más estable de los gradientes, especialmente en ciertas orientaciones. Gradiente Orientación muestra la dirección de los cambios de intensidad; en esta imagen, los colores representan distintas orientaciones de borde, lo que ayuda a identificar patrones estructurales del diseño del billete. La imagen FFT Magnitude (frecuencias) corresponde al dominio de la frecuencia y representa cómo se distribuyen las componentes espaciales de la imagen. Esta transformación permite observar regularidades globales, repeticiones de patrón y estructuras periódicas de impresión.

- En la tercera fila, FFT Color (patrones cromáticos) traslada la información de color al dominio frecuencial, permitiendo analizar la distribución repetitiva de patrones cromáticos y posibles irregularidades en la impresión. La representación LBP Textura (patrones locales) extrae patrones binarios locales, describiendo la textura de la superficie a partir de la relación entre cada píxel y sus vecinos. Este descriptor es útil para diferenciar superficies impresas con distinta calidad o granulado. La imagen Mapa Entropía Local representa el nivel de desorden o variabilidad en pequeñas regiones de la imagen; valores más altos suelen indicar zonas con mayor complejidad visual, detalle

o ruido. El Mapa Color Bordes resalta la concentración y respuesta de bordes mediante una escala de colores, mostrando qué regiones presentan mayor actividad estructural. Finalmente, Top Hat (detalles brillantes) extrae pequeños elementos claros sobre un fondo más oscuro, lo que ayuda a destacar brillos locales, microdetalles y zonas de alta reflectancia.

- En la cuarta fila, Black Hat (detalles oscuros) resalta pequeños detalles oscuros sobre fondos claros, permitiendo observar trazos finos, sombras locales o imperfecciones en zonas impresas. La imagen Adaptativo Gaussiano corresponde a una binarización adaptativa, donde el umbral cambia según la región de la imagen; esto facilita la separación de texto, patrones y contornos cuando la iluminación no es uniforme. La representación Binarización Otsu realiza una segmentación automática en blanco y negro a partir de un umbral global óptimo, útil para separar primer plano y fondo de manera simplificada. La imagen Detector ORB (puntos clave) muestra los puntos de interés detectados sobre el billete; estos puntos suelen ubicarse en esquinas, curvas y zonas ricas en detalle, siendo útiles para reconocimiento y comparación entre billetes. Por último, Gradiente Morfológico destaca el contorno de los objetos mediante operaciones morfológicas, resaltando los límites de números, retratos, textos y marcos.

De manera general, la figura 28 evidencia que la detección de falsificaciones no depende únicamente de la imagen original, sino del análisis complementario de múltiples representaciones. Algunas transformaciones son más adecuadas para estudiar el contraste y la nitidez; otras resaltan bordes, orientación y estructura; mientras que algunas permiten evaluar textura, frecuencia espacial o irregularidades estadísticas

La figura 29 presenta la aplicación de diferentes técnicas de suavizado sobre la imagen de un billete peruano, con el propósito de reducir ruido y atenuar variaciones indeseadas antes de la extracción de características.

Figura 29

Filtros de suavizado - mejora de ruido



Nota. Elaboración propia

En procesamiento digital de imágenes, el suavizado constituye una etapa preliminar orientada a mejorar la calidad visual de la imagen y a facilitar procesos posteriores como la detección de bordes, segmentación y clasificación. De manera general, una imagen degradada por ruido puede modelarse como:

$$g(x,y) = f(x,y) + \eta(x,y)$$

donde $f(x,y)$ representa la imagen ideal, $\eta(x,y)$ corresponde al ruido introducido durante la captura o transmisión, y $g(x,y)$ es la imagen observada. El objetivo de los filtros de suavizado es obtener una estimación $\hat{f}(x,y)$ lo más cercana posible a la imagen original, reduciendo el efecto del término de ruido.

La primera subfigura, denominada Original, muestra la imagen de entrada sin procesamiento. En ella todavía se observan pequeñas variaciones locales, ligeras irregularidades y textura visual del entorno de captura. Esta imagen sirve como referencia para evaluar el efecto de los distintos métodos de suavizado implementados. La subfigura Media 5x5 corresponde a la aplicación de un filtro promedio con una ventana de 5×5. Este filtro calcula el nuevo valor de cada píxel como el promedio aritmético de los valores contenidos en su vecindad. Su formulación general es:

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{i=-a}^{a} \sum_{j=-b}^{b} g\,(x+i, y+j)$$

donde $m \times n$ es el tamaño de la ventana, con $a = \frac{m-1}{2}$ y $b = \frac{n-1}{2}$. Para el caso 5×5, se promedian 25 valores. En la figura se aprecia una reducción moderada del ruido, aunque también se observa una pérdida parcial de nitidez en detalles finos del billete.

La subfigura Media 9x9 utiliza el mismo principio, pero con una ventana mayor. En este caso, el promedio se calcula sobre 81 píxeles vecinos, lo que incrementa la capacidad de suavizado. Sin embargo, al aumentar el tamaño de la máscara, también se incrementa el desenfoque. Por ello, en esta representación el billete aparece más difuso que en el caso anterior, con contornos menos definidos y menor preservación de detalles gráficos. La subfigura Gaussiano 5x5 muestra el resultado del filtrado gaussiano, el cual realiza un promedio ponderado donde los píxeles más cercanos al centro de la ventana tienen mayor peso que los más alejados. Su núcleo se define mediante la función:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

y la imagen suavizada se obtiene por convolución:

$$\hat{f}(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} G\,(i,j)\; g(x+i, y+j)$$

Este filtro produce una transición más natural que el filtro de media, ya que evita cambios bruscos y reduce ruido con menor generación de artefactos. En la figura, el resultado conserva mejor la estructura general del billete, aunque con una disminución controlada de la nitidez.

La subfigura Gaussiano 9x9 amplía el tamaño del núcleo gaussiano, produciendo un efecto de suavizado más intenso. En esta representación se reduce aún más la variación local, pero a costa de una mayor pérdida de detalle en los rasgos faciales, números y textos del billete. Esto evidencia que un incremento excesivo del tamaño del kernel puede afectar negativamente la información discriminante necesaria para la clasificación. La subfigura Mediana 5x5 aplica un filtro no lineal, en el cual el valor de salida no es un promedio, sino la mediana de los valores del vecindario. Su expresión puede representarse como:

$$\hat{f}(x,y) = \text{mediana}\{g(s,t) \mid (s,t) \in \mathcal{N}_{x,y}\}$$

donde $\mathcal{N}_{x,y}$ es la vecindad centrada en (x,y). Este filtro es especialmente útil frente al ruido impulsivo, ya que elimina valores atípicos sin difuminar tanto los bordes. En la figura, la imagen conserva mejor las estructuras principales del billete en comparación con los filtros promedio, mostrando un equilibrio adecuado entre reducción de ruido y preservación de contornos.

La subfigura Bilateral corresponde a un filtro que combina proximidad espacial y similitud radiométrica. A diferencia de los filtros lineales convencionales, este método suaviza regiones homogéneas sin degradar de forma significativa los bordes. Su formulación general es:

$$\hat{f}(p) = \frac{1}{W_p} \sum_{q \in \Omega} g\,(q) \exp\left(-\frac{\| p-q \|^2}{2\sigma_s^2}\right) \exp\left(-\frac{|g(p)-g(q)|^2}{2\sigma_r^2}\right)$$

donde p y q son posiciones de píxeles, σ_s controla la cercanía espacial, σ_r regula la similitud de intensidad y W_p es el factor de normalización. En la imagen obtenida se aprecia que el billete mantiene mejor sus bordes y elementos relevantes, mientras se atenúan pequeñas fluctuaciones de ruido.

La última subfigura, Non-Local Means, emplea una estrategia de suavizado basada en la similitud entre parches de la imagen, no solo entre vecinos inmediatos. Este método estima el valor de cada píxel a partir de una combinación ponderada de otros píxeles con vecindarios semejantes. Su expresión general es:

$$\hat{f}(p) = \sum_{q \in \Omega} w\,(p, q)\, g(q)$$

con pesos definidos por:

$$w(p, q) = \frac{1}{Z(p)} \exp\left(- \frac{\| P(p) - P(q) \|_2^2}{h^2} \right)$$

donde $P(p)$ y $P(q)$ representan parches centrados en p y q, h controla el nivel de filtrado y $Z(p)$ es una constante de normalización. Este enfoque tiende a preservar estructuras repetitivas y patrones texturales. En la figura, se observa una imagen más homogénea, aunque con cierto efecto de suavizado extendido que puede disminuir detalles finos si los parámetros no son ajustados cuidadosamente.

En conjunto, la figura 29 evidencia que cada filtro produce un comportamiento distinto frente al ruido y la preservación de detalles. Los filtros de media y gaussiano reducen ruido de forma progresiva, pero generan mayor desenfoque cuando aumenta el tamaño de la máscara. El filtro de mediana mejora la robustez ante ruido impulsivo y conserva mejor las transiciones. El filtro bilateral mantiene bordes más definidos, mientras que Non-Local Means ofrece una reducción más sofisticada a partir de similitudes globales en la imagen. En el contexto de esta investigación, estas técnicas permiten determinar qué método de suavizado ofrece el mejor equilibrio entre limpieza visual y conservación de rasgos distintivos del billete, lo cual resulta conveniente para las etapas posteriores de extracción de características y clasificación entre billetes auténticos y falsificados.

La figura 30 muestra distintas técnicas de mejora de contraste aplicadas a la imagen de un billete peruano, con el propósito de resaltar detalles visuales que pueden resultar útiles en la detección de falsificaciones. En procesamiento digital de imágenes, el contraste expresa la diferencia de intensidad entre regiones claras y oscuras.

Figura 30

Mejora de contraste - realce de detalles



Nota. Elaboración propia

Cuando una imagen presenta bajo contraste, algunos elementos del billete, como textos, números, tramas, relieves aparentes o patrones de seguridad, pueden quedar poco definidos. Por ello, se emplean transformaciones puntuales y métodos basados en histogramas para redistribuir los niveles de intensidad y hacer más visibles dichos detalles.

Sea r la intensidad de entrada y s la intensidad de salida, una mejora de contraste puede representarse en forma general mediante:

$$s = T(r)$$

donde $T(\cdot)$ es una transformación que modifica la distribución tonal de la imagen.

La subfigura Original corresponde a la imagen de entrada sin realce. En ella se aprecia el billete con contraste moderado y una distribución tonal limitada por las condiciones de captura. Esta imagen constituye la referencia para evaluar el efecto de cada técnica.

La subfigura Ecualización de histograma aplica una transformación global basada en la función de distribución acumulada del histograma. Su objetivo es expandir los niveles de gris o intensidad sobre todo el rango dinámico disponible. Matemáticamente, para un nivel r_k, la transformación puede expresarse como:

$$s_k = (L-1)\sum_{j=0}^{k} p_r\left(r_j\right)$$

donde L es el número total de niveles de intensidad y $p_r(r_j)$ es la probabilidad de ocurrencia del nivel r_j. En la figura, esta técnica incrementa el contraste global del billete, haciendo más notorios el retrato, la numeración y los bordes principales. Sin embargo, también puede intensificar zonas ya contrastadas y producir una apariencia más rígida o artificial en algunas regiones.

Las subfiguras CLAHE clip = 2 y CLAHE clip = 4 corresponden a la ecualización adaptativa limitada por contraste (*Contrast Limited Adaptive Histogram Equalization*). A diferencia de la ecualización global, este método trabaja por bloques locales o teselas, aplicando la ecualización en cada región y luego interpolando los resultados. De forma local, puede representarse como:

$$s_T(x,y) = (L-1)\,CDF_T\big(r(x,y)\big)$$

donde CDF_T es la función de distribución acumulada calculada en la tesela T. El parámetro *clip* limita la amplificación excesiva del histograma local, evitando que el ruido sea reforzado de manera desproporcionada. En la imagen con clip = 2 se observa un realce moderado y más controlado de las texturas y contornos del billete. En cambio, con clip = 4 el incremento del contraste local es más pronunciado, permitiendo resaltar con mayor intensidad

líneas finas, zonas impresas y variaciones de textura, aunque con el riesgo de exagerar ciertas irregularidades visuales.

Las subfiguras Gamma 0.5, Gamma 1.5 y Gamma 2.0 muestran el efecto de la corrección gamma, que consiste en una transformación no lineal de la intensidad. Su expresión general es:

$$s = c\, r^{\gamma}$$

donde c es una constante de normalización y γ controla la forma de la curva tonal. Esta técnica modifica la luminancia de manera selectiva, afectando en distinta medida las zonas claras y oscuras. En la figura 30, se aprecia que la variación del parámetro gamma altera progresivamente la claridad del billete, generando imágenes más oscuras o más claras según la configuración empleada en la implementación. Esto permite estudiar cómo cambian la visibilidad de elementos como el rostro, el número de denominación, las tramas de fondo y los bordes de impresión bajo distintas condiciones de realce tonal.

La subfigura Stretching corresponde al estiramiento lineal de contraste. Este método toma el rango mínimo y máximo de intensidades presentes en la imagen y lo expande al rango dinámico completo. Su formulación es:

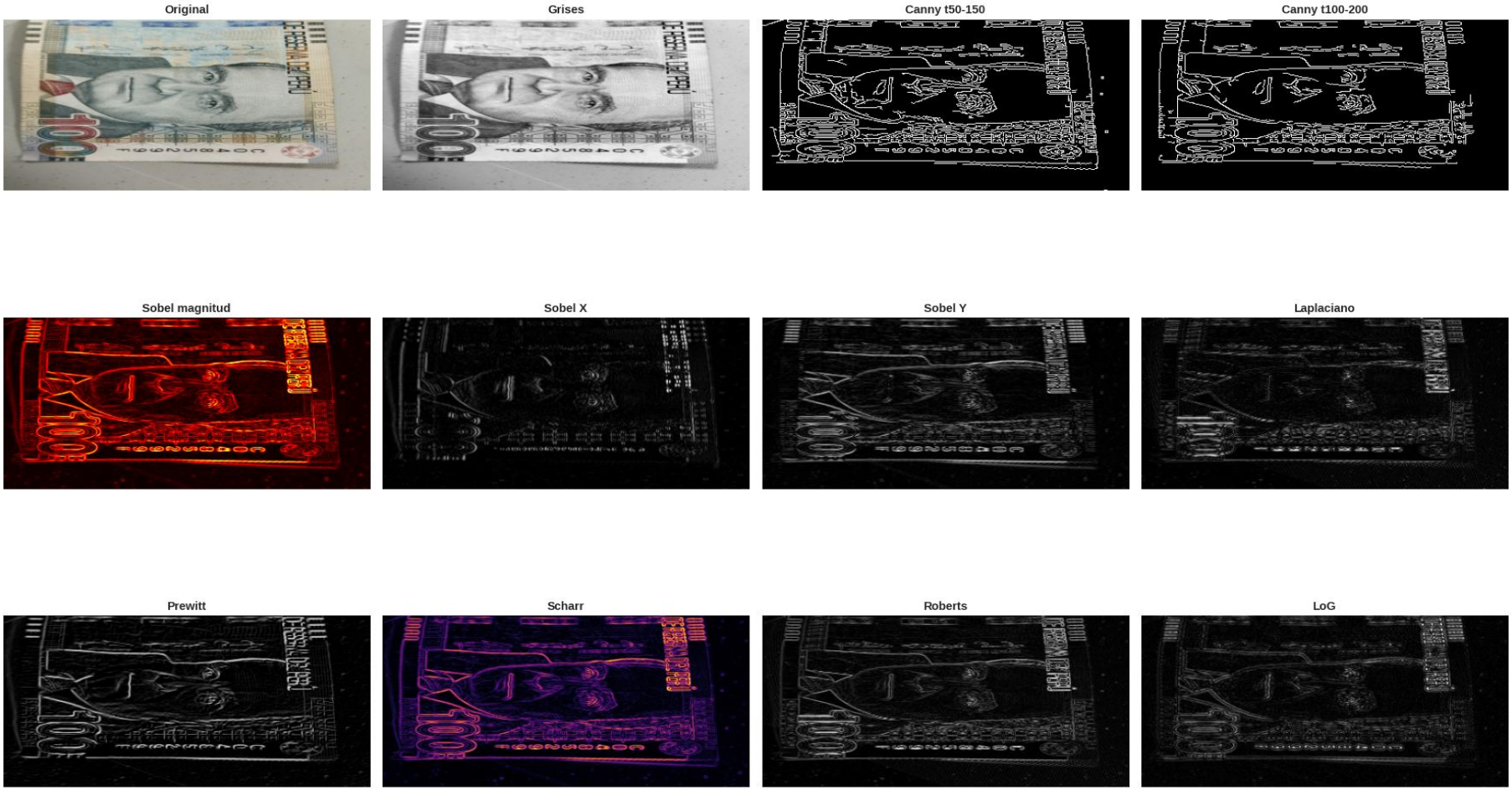
$$s = \frac{r - r_{\min}}{r_{\max} - r_{\min}} (L - 1)$$

donde $r_{\min}$ y $r_{\max}$ representan las intensidades mínima y máxima de la imagen original. En la figura, esta transformación mejora la separación entre zonas claras y oscuras sin modificar de forma tan agresiva la estructura tonal, por lo que permite observar con mayor claridad contornos, áreas de sombreado y componentes cromáticos del billete. En conjunto, la figura 30 evidencia que cada técnica de realce modifica de forma distinta la apariencia del billete. La ecualización de histograma mejora el contraste global; CLAHE favorece el realce local de detalles; la corrección gamma ajusta de manera no lineal la luminancia; y el *stretching* redistribuye linealmente el rango tonal.

La figura 31 presenta distintas técnicas de detección de bordes aplicadas a la imagen de un billete peruano, con el propósito de resaltar contornos, transiciones de intensidad y estructuras gráficas que pueden contribuir al proceso de identificación de falsificaciones.

Figura 31

Detección de bordes



Nota. Elaboración propia

De manera general, si una imagen en escala de grises se representa como $I(x, y)$, los bordes se localizan a partir de cambios espaciales de intensidad, los cuales pueden medirse mediante derivadas de primer o segundo orden. El gradiente bidimensional se expresa como:

$$\nabla I(x, y) = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

y su magnitud se calcula mediante:

$$|\nabla I(x, y)| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

Esta magnitud indica qué tan fuerte es el cambio de intensidad en cada punto de la imagen.

La subfigura Original muestra la imagen RGB del billete sin procesamiento, mientras que la subfigura Grises corresponde a su conversión a escala de grises. Esta transformación reduce la imagen de tres canales a uno solo, simplificando el análisis de intensidad. En términos generales, la conversión puede expresarse como una combinación ponderada de los canales rojo, verde y azul:

$$I_{gris} = 0.299R + 0.587G + 0.114B$$

A partir de esta representación se aplican los diferentes operadores de detección de bordes.

La subfigura Canny t50-150 muestra el resultado del detector de Canny con umbrales bajo y alto de 50 y 150, respectivamente. Este método comprende suavizado gaussiano, cálculo del gradiente, supresión de no máximos y umbralización por histéresis. Suavizado inicial:

$$I_s = I * G_\sigma$$

donde $*$ representa la convolución y G_σ es el filtro gaussiano. Luego, la magnitud del gradiente se compara con dos umbrales T_b y T_a. Si:

$$|\nabla I| \geq T_a$$

el píxel se considera borde fuerte; si

$$T_b \leq |\nabla I| < T_a$$

se considera borde débil y solo se conserva si está conectado a un borde fuerte. En la imagen se observa una gran cantidad de contornos del retrato, números, textos y marcos del billete.

La subfigura Canny 100-200 aplica el mismo procedimiento, pero con umbrales mayores. Esto reduce la cantidad de bordes detectados, conservando únicamente las

transiciones más marcadas. En consecuencia, se obtiene una representación más selectiva y menos sensible a pequeñas variaciones o ruido residual.

La subfigura Sobel magnitud representa la magnitud del gradiente obtenida a partir de los operadores Sobel en las direcciones horizontal y vertical. Sus máscaras usuales son:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Con ellas se calculan:

$$S_x = I * G_x, \qquad S_y = I * G_y$$

y posteriormente:

$$M_{Sobel} = \sqrt{S_x^2 + S_y^2}$$

En la figura, esta representación resalta con intensidad elevada los contornos principales del billete, como el rostro, la denominación y los límites externos.

La subfigura Sobel X muestra únicamente la respuesta del gradiente en dirección horizontal, es decir, la sensibilidad a cambios de intensidad a lo largo del eje x. Esta salida resalta principalmente bordes verticales presentes en la imagen, como textos laterales y marcos. Por su parte, Sobel Y representa el gradiente en dirección vertical, destacando sobre todo bordes horizontales, como líneas decorativas y separaciones tonales del diseño del billete.

La subfigura Laplaciano corresponde a un operador de segunda derivada, utilizado para detectar regiones donde la intensidad cambia bruscamente en cualquier dirección. Matemáticamente, se define como:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Una aproximación discreta común es:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

o variantes equivalentes. En la figura 31, este operador enfatiza zonas de transición intensa, aunque también puede ser más sensible al ruido que los operadores basados en gradiente.

La subfigura Prewitt emplea máscaras similares a Sobel, pero con ponderaciones uniformes. Sus núcleos habituales son:

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \qquad P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

El cálculo de la magnitud sigue la misma lógica:

$$M_{Prewitt} = \sqrt{(I * P_x)^2 + \left(I * P_y\right)^2}$$

En la imagen, este método produce un mapa de bordes claro y continuo, útil para identificar contornos generales del billete sin un realce tan agresivo como otros operadores.

La subfigura Scharr mejora la estimación del gradiente, especialmente en términos de simetría rotacional y precisión en la orientación de los bordes. Sus máscaras son:

$$Sch_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}, \qquad Sch_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

La magnitud se obtiene de forma análoga:

$$M_{Scharr} = \sqrt{(I * Sch_x)^2 + \left(I * Sch_y\right)^2}$$

En la figura, este operador destaca con mayor claridad ciertos bordes del retrato, textos y numeración, mostrando una respuesta más intensa en estructuras relevantes.

La subfigura Roberts aplica un operador basado en diferencias diagonales. Sus máscaras son pequeñas, de tamaño 2×2:

$$R_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad R_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Su magnitud puede expresarse como:

$$M_{Roberts} = \sqrt{(I * R_x)^2 + \left(I * R_y\right)^2}$$

Este método responde con rapidez a transiciones locales, aunque resulta más sensible al ruido y suele generar bordes más finos. En la figura, resalta contornos esenciales del billete, pero con una respuesta menos uniforme que Sobel o Scharr.

Finalmente, la subfigura LoG corresponde al operador *Laplacian of Gaussian*, que combina un suavizado gaussiano previo con la aplicación del Laplaciano. Su formulación general es:

$$LoG(x, y) = \nabla^2\big(G_\sigma(x, y) * I(x, y)\big)$$

o de forma equivalente, aplicando primero el filtro gaussiano y luego la segunda derivada. Este método reduce parcialmente la sensibilidad al ruido antes de detectar cambios bruscos de intensidad. En la figura, el resultado muestra bordes más controlados que el Laplaciano directo, permitiendo distinguir estructuras del billete con una respuesta menos irregular.

En conjunto, la figura 31 evidencia que cada operador de bordes resalta propiedades distintas de la imagen. Los métodos basados en gradiente, como Sobel, Prewitt y Scharr,

permiten identificar dirección e intensidad de los cambios. Canny produce bordes binarios más definidos mediante histéresis. El Laplaciano y LoG resaltan transiciones de segundo orden, mientras que Roberts detecta cambios diagonales locales.

La figura 32 presenta la aplicación de distintas operaciones morfológicas sobre la imagen en escala de grises de un billete peruano, con el propósito de resaltar estructuras geométricas, contornos, detalles locales y variaciones de intensidad asociadas a la forma de los elementos impresos. En procesamiento digital de imágenes, la morfología matemática permite transformar la imagen a partir de un elemento estructurante B, el cual actúa como una plantilla local para analizar la distribución espacial de los píxeles. Este tipo de procesamiento resulta útil cuando se desea estudiar la forma de números, textos, marcos, retratos, líneas ornamentales y otros componentes visuales del billete.

Sea $f(x, y)$ la imagen de entrada y B el elemento estructurante. Las operaciones morfológicas modifican la imagen según relaciones de vecindad, máximos y mínimos locales. En este contexto, la primera subfigura, Original (grises), corresponde a la imagen de referencia sobre la cual se aplican las transformaciones posteriores. En ella se observan el retrato, la denominación, los textos y las áreas de fondo con sus variaciones de intensidad originales.

La subfigura Erosión reduce las regiones claras de la imagen y amplía relativamente las oscuras. En imágenes en escala de grises, la erosión puede expresarse como:

$$(f \ominus B)(x, y) = \min_{(s,t)\in B} \{f(x + s, y + t) - B(s, t)\}$$

Si el elemento estructurante es plano, la expresión se simplifica al mínimo local del vecindario. Esta operación atenúa pequeños detalles brillantes y contrae contornos claros, por lo que en la figura se aprecia una disminución del grosor de ciertas zonas del billete y una mayor presencia visual de áreas oscuras.

La subfigura Dilatación realiza el efecto opuesto: expande regiones claras y reduce relativamente las oscuras. Matemáticamente, se define como:

$$(f \oplus B)(x, y) = \max_{(s,t)\in B} \{f(x - s, y - t) + B(s, t)\}$$

Para un elemento estructurante plano, la operación equivale al máximo local. En la figura, esta transformación produce una expansión de las zonas brillantes del billete, lo que hace que algunos bordes y áreas claras se vean más anchas o suavizadas.

La subfigura Apertura corresponde a la combinación sucesiva de erosión y dilatación:

$$f \circ B = (f \ominus B) \oplus B$$

Esta operación elimina pequeñas estructuras brillantes aisladas, suaviza contornos y preserva mejor las formas generales de mayor tamaño. En la imagen se observa una reducción de pequeñas irregularidades luminosas, mientras se mantiene la estructura principal del retrato y de la numeración del billete. La apertura resulta útil cuando se busca disminuir ruido claro o detalles finos poco relevantes.

La subfigura Cierre se define como la aplicación de dilatación seguida de erosión:

$$f \bullet B = (f \oplus B) \ominus B$$

Su efecto consiste en rellenar pequeñas discontinuidades oscuras, cerrar huecos y suavizar contornos desde una perspectiva complementaria a la apertura. En la figura, el cierre genera una apariencia más compacta en ciertas regiones del billete, reduciendo pequeñas interrupciones en líneas y zonas impresas.

La subfigura Gradiente Morfológico resalta los contornos mediante la diferencia entre la dilatación y la erosión. Su formulación es:

$$G_m(f) = (f \oplus B) - (f \ominus B)$$

Esta operación enfatiza las transiciones de intensidad ubicadas en los límites de las estructuras. En la figura, el resultado muestra con mayor nitidez los contornos del retrato, la denominación “100”, los textos verticales y otras formas impresas. Por ello, el gradiente morfológico es útil para el análisis de bordes desde una perspectiva basada en forma local.

La subfigura Top Hat representa la diferencia entre la imagen original y su apertura:

$$T_h(f) = f - (f \circ B)$$

Esta transformación permite extraer pequeños detalles brillantes que fueron eliminados por la apertura. En la figura, el Top Hat resalta elementos claros de reducido tamaño presentes en el billete, como reflejos locales, zonas de mayor intensidad y pequeñas estructuras impresas. Su uso es conveniente cuando se desea identificar detalles finos claros que podrían contener información distintiva de autenticidad o deterioro.

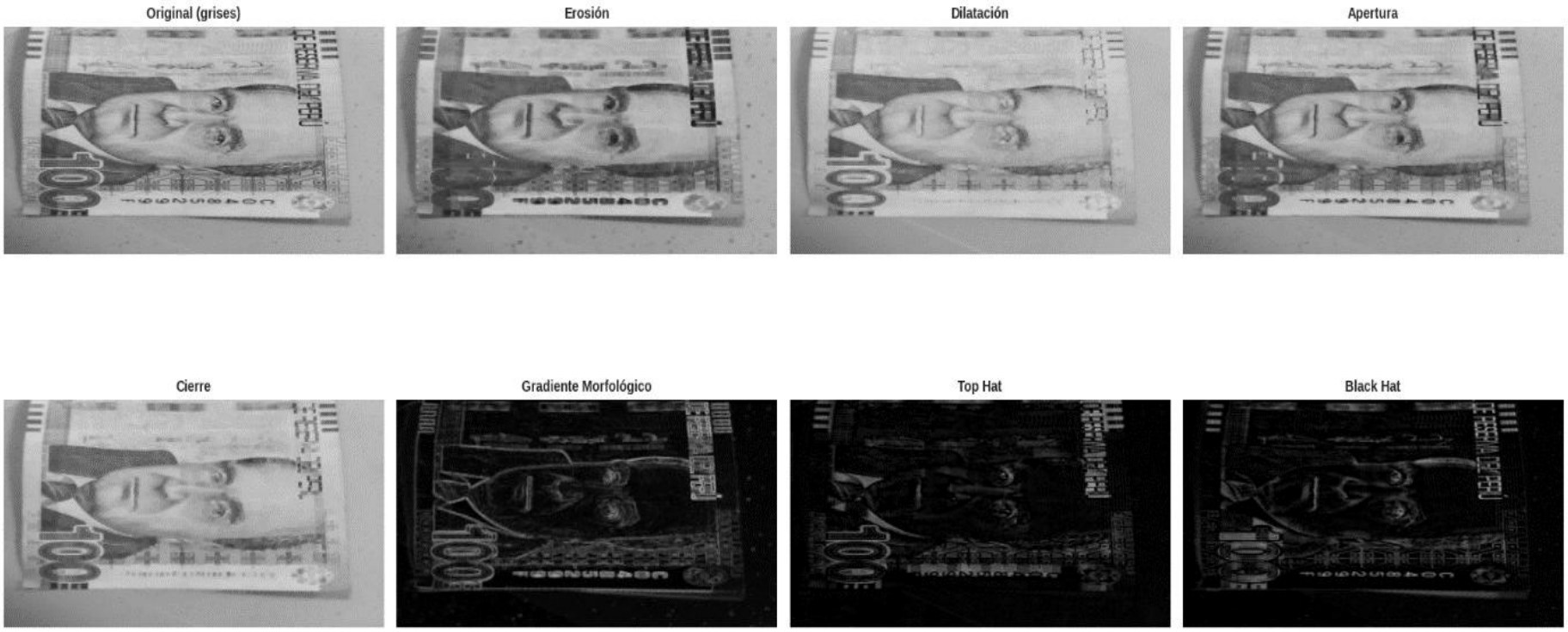
La subfigura Black Hat corresponde a la diferencia entre el cierre y la imagen original:

$$B_h(f) = (f \bullet B) - f$$

Esta operación resalta pequeños detalles oscuros sobre fondos relativamente más claros. En la figura, el Black Hat pone en evidencia líneas oscuras del retrato, contornos de letras, trazos finos y pequeñas variaciones tonales asociadas al diseño del billete. Esta representación puede aportar información sobre irregularidades de impresión, sombras locales o discontinuidades presentes en la superficie observada.

Figura 32

Operaciones morfológicas - análisis de forma



Nota. Elaboración propia

En conjunto, la figura 32 muestra que las operaciones morfológicas permiten estudiar la imagen del billete desde el punto de vista de su forma, sus contornos y sus detalles locales. La erosión y la dilatación modifican la extensión de regiones claras y oscuras; la apertura y el cierre permiten filtrar estructuras pequeñas; el gradiente morfológico enfatiza límites; y las transformaciones Top Hat y Black Hat aíslan detalles brillantes y oscuros, respectivamente.

4.3. Arquitecturas CNN y CNN-RNN

Con la finalidad de evaluar distintas estrategias de extracción y modelado de características visuales en billetes peruanos, se diseñaron tres arquitecturas de aprendizaje profundo: una arquitectura CNN pura, una arquitectura CNN-LSTM bidireccional y una arquitectura CNN-GRU bidireccional. Todas parten de imágenes de entrada de tamaño $128 \times 128 \times 3$ y generan una salida multiclase de 16 categorías. La diferencia entre ellas radica en la forma en que procesan la información extraída: mientras la CNN pura realiza la clasificación directamente sobre descriptores espaciales, las arquitecturas híbridas CNN-RNN incorporan capas recurrentes para modelar dependencias secuenciales en las características obtenidas por la etapa convolucional.

La Tabla 3 presenta la primera arquitectura propuesta, denominada CNN_Billetes, la cual corresponde a una red neuronal convolucional profunda orientada al aprendizaje de patrones espaciales presentes en las imágenes de billetes. Esta arquitectura recibe como entrada una imagen RGB de dimensión $128 \times 128 \times 3$, sobre la cual se aplican bloques sucesivos de convolución, normalización por lotes, agrupamiento máximo y regularización por *dropout*.

La operación central de una capa convolucional puede representarse mediante:

$$x_{i,j,k}^{(l)} = \phi\left(\sum_{m}\sum_{u}\sum_{v} w_{u,v,m,k}^{(l)}\; x_{i+u,j+v,m}^{(l-1)} + b_k^{(l)}\right)$$

donde $x_{i,j,k}^{(l)}$ es la activación de la capa l en la posición (i, j) y canal k, w representa los pesos del filtro, $b_k^{(l)}$ es el sesgo y $\phi(\cdot)$ corresponde a la función de activación. Esta operación permite detectar bordes, texturas, contornos y patrones de impresión del billete.

La arquitectura CNN pura está compuesta por cuatro etapas de extracción jerárquica. En la primera se emplean dos capas convolucionales de 32 filtros, seguidas de normalización y max pooling. En la segunda etapa se incrementa la profundidad a 64 filtros; en la tercera, a 128 filtros; y en la cuarta, a 256 filtros. Esta progresión permite que la red aprenda primero rasgos simples y luego estructuras más abstractas y discriminantes.

La capa de max pooling reduce la dimensionalidad espacial, conservando las respuestas más representativas. Su operación puede expresarse como:

$$y_{i,j,k} = \max_{(u,v)\in\Omega} x_{i+u,j+v,k}$$

donde Ω es la ventana local de agrupamiento. Esta reducción disminuye el costo computacional y aporta cierta invariancia ante pequeñas traslaciones.

La normalización por lotes estabiliza la distribución de activaciones en cada capa, favoreciendo un entrenamiento más robusto. De manera general, se expresa como:

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \qquad y = \gamma\hat{x} + \beta$$

donde μ_B y σ_B^2 son la media y varianza del lote, mientras que γ y β son parámetros aprendibles. Después del último bloque convolucional, la arquitectura emplea una capa Global Average Pooling, que resume cada mapa de características en un solo valor promedio:

$$g_k = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_{i,j,k}$$

Esto produce un vector de 256 características, que luego es procesado por dos capas densas de 256 y 128 neuronas, respectivamente. Finalmente, la salida de clasificación se obtiene con una capa densa de 16 neuronas, asociada a las categorías del problema.

Desde el punto de vista estructural, esta arquitectura alcanza 685,744 parámetros totales, de los cuales 684,336 son entrenables. Esta mayor cantidad de parámetros le brinda alta capacidad de representación, aunque también implica mayor costo computacional y mayor riesgo de sobreajuste si no se controla adecuadamente el entrenamiento.

Tabla 3

Arquitectura 1: CNN Puro

Model: "CNN_Billetes"		
Layer (type)	**Output Shape**	**Param #**
entrada_imagen (InputLayer)	(None, 128, 128, 3)	0
conv1_1 (Conv2D)	(None, 128, 128, 32)	896
batch_normalization (BatchNormalization)	(None, 128, 128, 32)	128
conv1_2 (Conv2D)	(None, 128, 128, 32)	9,248
batch_normalization_1 (BatchNormalization)	(None, 128, 128, 32)	128
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
dropout (Dropout)	(None, 64, 64, 32)	0
conv2_1 (Conv2D)	(None, 64, 64, 64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 64, 64, 64)	256
conv2_2 (Conv2D)	(None, 64, 64, 64)	36,928
batch_normalization_3 (BatchNormalization)	(None, 64, 64, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)	0
dropout_1 (Dropout)	(None, 32, 32, 64)	0
conv3_1 (Conv2D)	(None, 32, 32, 128)	73,856
batch_normalization_4 (BatchNormalization)	(None, 32, 32, 128)	512
conv3_2 (Conv2D)	(None, 32, 32, 128)	147,584
batch_normalization_5 (BatchNormalization)	(None, 32, 32, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 128)	0
dropout_2 (Dropout)	(None, 16, 16, 128)	0
conv4_1 (Conv2D)	(None, 16, 16, 256)	295,168
batch_normalization_6 (BatchNormalization)	(None, 16, 16, 256)	1,024
global_average_pooling2d (GlobalAveragePooling2D)	(None, 256)	0
dense (Dense)	(None, 256)	65,792
dropout_3 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dropout_4 (Dropout)	(None, 128)	0
clasificacion (Dense)	(None, 16)	2,064
Total params:	**685,744**	*2.62 MB*
Trainable params:	**684,336**	*2.61 MB*
Non-trainable params:	**1,408**	*5.50 KB*

Nota. Elaboración propia

La Tabla 4 presenta la segunda arquitectura, denominada CNN_LSTM_Hibrido, la cual combina una etapa convolucional con una etapa recurrente basada en LSTM bidireccional. En este caso, la CNN actúa como extractor de características espaciales, mientras que la LSTM modela relaciones secuenciales dentro de la representación interna generada por los mapas de activación.

Tabla 4

Arquitectura 2: CNN-LSTM Bidireccional

Model: "CNN_LSTM_Hibrido"		
Layer (type)	**Output Shape**	**Param #**
entrada_imagen (InputLayer)	(None, 128, 128, 3)	0
conv2d (Conv2D)	(None, 128, 128, 32)	896
batch_normalization_7 (BatchNormalization)	(None, 128, 128, 32)	128
max_pooling2d_3 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_1 (Conv2D)	(None, 64, 64, 64)	18,496
batch_normalization_8 (BatchNormalization)	(None, 64, 64, 64)	256
max_pooling2d_4 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_2 (Conv2D)	(None, 32, 32, 128)	73,856
batch_normalization_9 (BatchNormalization)	(None, 32, 32, 128)	512
max_pooling2d_5 (MaxPooling2D)	(None, 16, 16, 128)	0
reshape (Reshape)	(None, 256, 128)	0
bidirectional (Bidirectional)	(None, 256, 128)	98,816
dropout_5 (Dropout)	(None, 256, 128)	0
bidirectional_1 (Bidirectional)	(None, 64)	41,216
dropout_6 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 128)	8,320
dropout_7 (Dropout)	(None, 128)	0
clasificacion (Dense)	(None, 16)	2,064
Total params:	**244,560**	*0.93 MB*
Trainable params:	**244,112**	*0.93 MB*
Non-trainable params:	**448**	*1.75 KB*

Nota. Elaboración propia

La primera parte de esta arquitectura incluye tres bloques convolucionales con 32, 64 y 128 filtros, cada uno acompañado de batch normalization y max pooling. Luego del último bloque, se obtiene una salida de dimensión $(16,16,128)$. Esta representación es reorganizada

mediante una capa Reshape hacia una secuencia de tamaño $(256,128)$, donde 256 corresponde al número de pasos temporales y 128 al número de características por paso.

La unidad LSTM está diseñada para modelar dependencias de largo alcance mediante compuertas de entrada, olvido y salida. Sus ecuaciones principales son:

$$f_t = \sigma\big(W_f[h_{t-1}, x_t] + b_f\big) \quad i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \; o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \; h_t = o_t \odot \tanh(C_t)$$

donde x_t es la entrada en el tiempo t, h_t es el estado oculto, C_t es el estado de memoria y σ representa la función sigmoide.

El carácter bidireccional implica que la secuencia se procesa tanto en sentido directo como inverso, y luego ambas salidas se concatenan:

$$h_t^{(bi)} = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]$$

Esto permite que la red utilice información contextual de ambos sentidos, lo cual resulta útil cuando la representación espacial puede interpretarse como una secuencia de patrones relacionados. La arquitectura CNN-LSTM incluye dos capas bidireccionales: la primera produce una salida $(256,128)$ y la segunda reduce la representación a un vector de 64 unidades. Posteriormente, una capa densa de 128 neuronas y una capa de clasificación de 16 salidas completan el modelo.

Esta arquitectura presenta 244,560 parámetros totales, con 244,112 parámetros entrenables. En comparación con la CNN pura, reduce notablemente la complejidad del modelo al conservar una etapa convolucional más compacta y transferir parte del procesamiento a la memoria recurrente.

La Tabla 5 muestra la tercera arquitectura, denominada CNN_GRU_Hibrido, la cual mantiene una estructura similar a la anterior, pero reemplaza las unidades LSTM por unidades GRU bidireccionales. La lógica general sigue siendo la misma: la CNN extrae descriptores espaciales y la GRU modela dependencias secuenciales dentro de esa representación. La fase convolucional repite tres bloques con 32, 64 y 128 filtros, seguidos de batch normalization y max pooling. Tras ello, la salida (16,16,128) es reordenada nuevamente en una secuencia (256,128), que luego se procesa con dos capas bidireccionales GRU.

Tabla 5

Arquitectura 3: CNN-GRU Bidireccional

Model: "CNN_GRU_Hibrido"		
Layer (type)	**Output Shape**	**Param #**
entrada_imagen (InputLayer)	(None, 128, 128, 3)	0
conv2d_3 (Conv2D)	(None, 128, 128, 32)	896
batch_normalization_10 (BatchNormalization)	(None, 128, 128, 32)	128
max_pooling2d_6 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_4 (Conv2D)	(None, 64, 64, 64)	18,496
batch_normalization_11 (BatchNormalization)	(None, 64, 64, 64)	256
max_pooling2d_7 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_5 (Conv2D)	(None, 32, 32, 128)	73,856
batch_normalization_12 (BatchNormalization)	(None, 32, 32, 128)	512
max_pooling2d_8 (MaxPooling2D)	(None, 16, 16, 128)	0
reshape_1 (Reshape)	(None, 256, 128)	0
bidirectional_2 (Bidirectional)	(None, 256, 128)	74,496
dropout_8 (Dropout)	(None, 256, 128)	0
bidirectional_3 (Bidirectional)	(None, 64)	31,104
dropout_9 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 128)	8,320
dropout_10 (Dropout)	(None, 128)	0
clasificacion (Dense)	(None, 16)	2,064
Total params:	**210,128**	*0.80 MB*
Trainable params:	**209,680**	*0.80 MB*
Non-trainable params:	**448**	*1.75 KB*

Nota. Elaboración propia

La unidad GRU presenta una estructura más compacta que la LSTM, ya que utiliza solo dos compuertas principales: actualización y reinicio. Sus ecuaciones se expresan como:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z)\ r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)\ \tilde{h}_t = \tanh(W_h[r_t \odot h_{t-1}, x_t] + b_h)\ h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

donde z_t regula cuánto de la información previa se conserva y r_t controla cuánto del estado anterior participa en la generación del nuevo contenido. Al igual que en el caso LSTM, el procesamiento bidireccional concatena la información obtenida en ambos sentidos.

La ventaja de la GRU radica en su menor número de parámetros y en una dinámica de entrenamiento generalmente más ligera. En esta arquitectura, la primera capa bidireccional produce una salida (256,128), mientras que la segunda reduce la representación a 64 unidades. Luego se aplica una capa densa de 128 neuronas y la capa final de clasificación de 16 categorías. El modelo CNN-GRU bidireccional alcanza 210,128 parámetros totales, con 209,680 entrenables, lo que lo convierte en la arquitectura más compacta de las tres analizadas.

La Tabla 6 resume la comparación global de parámetros entre los tres modelos propuestos. La arquitectura CNN pura es la de mayor complejidad, con 685,744 parámetros y un tamaño aproximado de 2.62 MB. La arquitectura CNN-LSTM bidireccional reduce esta cifra a 244,560 parámetros y 0.93 MB, mientras que la arquitectura CNN-GRU bidireccional alcanza el menor valor, con 210,128 parámetros y 0.80 MB.

Tabla 6

Comparativa de parámetros - Arquitecturas de Red Neuronal

Modelo	Parámetros Totales	Parámetros (MB)	Arquitectura
CNN Puro	685,744	2.62 MB	ARQUITECTURA 1: CNN Puro
CNN-LSTM Bidireccional	244,560	0.93 MB	ARQUITECTURA 2: CNN-LSTM Bidireccional
CNN-GRU Bidireccional	210,128	0.80 MB	ARQUITECTURA 3: CNN-GRU Bidireccional
CNN-GRU Bidireccional es el modelo más eficiente (menor cantidad de parámetros: 210,128 / 0.80 MB)			
DETALLE DE PARÁMETROS POR MODELO			
Modelo	**Total Params**	**Trainable Params**	**Non-trainable Params**
CNN Puro	685,744	684,336	1,408
CNN-LSTM Bidireccional	244,560	244,112	448
CNN-GRU Bidireccional	210,128	209,680	448

Nota. Elaboración propia

Desde la perspectiva de eficiencia estructural, el modelo CNN-LSTM reduce aproximadamente 64.34% de los parámetros respecto de la CNN pura, mientras que el modelo CNN-GRU reduce alrededor de 69.36% frente a esa misma arquitectura. Asimismo, la CNN-

GRU presenta cerca de 14.08% menos parámetros que la CNN-LSTM, lo que confirma su mayor compacidad estructural. Estos resultados permiten afirmar que las arquitecturas híbridas CNN-RNN ofrecen una alternativa más ligera que la CNN pura, al combinar extracción espacial con modelado secuencial sin requerir una gran cantidad de parámetros. En particular, la arquitectura CNN-GRU bidireccional destaca como la más eficiente en términos de tamaño y complejidad, mientras que la arquitectura CNN-LSTM bidireccional constituye una solución intermedia entre capacidad secuencial y costo computacional. Por su parte, la CNN pura conserva una mayor capacidad de representación espacial directa, aunque con una demanda más alta de memoria y procesamiento.

4.4. Entrenamiento de modelos

El entrenamiento de los modelos se realizó con el propósito de aprender patrones discriminantes asociados a billetes peruanos en distintas condiciones visuales. Para ello, las imágenes fueron organizadas, preprocesadas y presentadas a las arquitecturas propuestas en lotes sucesivos, con el fin de ajustar sus pesos internos mediante minimización del error de clasificación. En este proceso, la salida de cada modelo se obtiene mediante una función *softmax*, expresada como:

$$\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^{C} e^{z_j}}$$

donde $\hat{y}_k$ es la probabilidad estimada para la clase k, z_k corresponde a la activación de salida de dicha clase y C es el número total de categorías. A partir de estas probabilidades, la pérdida de entrenamiento para clasificación multiclase puede representarse mediante la entropía cruzada categórica:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik}\log(\hat{y}_{ik})$$

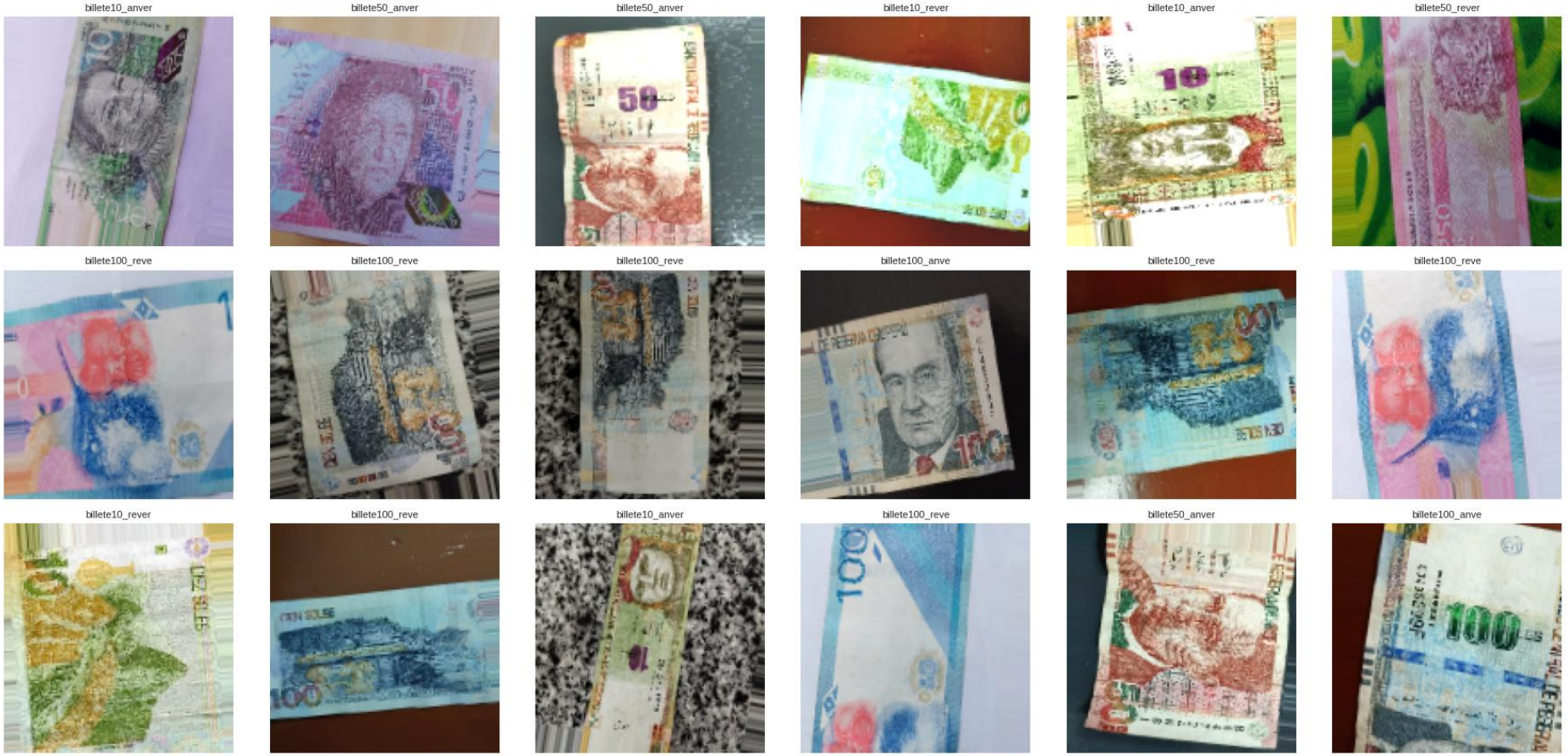
donde N es el número de muestras, y_{ik} es la etiqueta real y $\hat{y}_{ik}$ es la probabilidad predicha por el modelo. Asimismo, la exactitud se calculó como la proporción de predicciones correctas:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\,(\hat{c}_i = c_i)$$

donde $\hat{c}_i$ es la clase predicha y c_i es la clase real

Figura 33

Preparación de datos para entrenamiento



Nota. Elaboración propia

La figura 33 muestra una visualización representativa de las imágenes utilizadas durante la preparación de datos para el entrenamiento. En ella se observan billetes correspondientes a distintas denominaciones y orientaciones, tales como anverso y reverso, además de variaciones visibles en iluminación, inclinación, escala, recorte y fondo. Esta diversidad de muestras permite que el modelo no aprenda únicamente una representación rígida del billete, sino que desarrolle tolerancia frente a cambios geométricos y radiométricos presentes en escenarios de captura reales. Desde el punto de vista del aprendizaje profundo, esta etapa busca que la red procese ejemplos heterogéneos, conservando la información más representativa de cada clase. Dado que las arquitecturas propuestas reciben imágenes de tamaño 128×128×3, la preparación del conjunto también permite homogenizar la entrada y mantener consistencia dimensional en todas las muestras. En términos prácticos, esta figura evidencia que el entrenamiento no se realizó con imágenes perfectamente uniformes, sino con ejemplos que presentan diferentes condiciones visuales, lo cual resulta conveniente para evaluar la capacidad de generalización del sistema.

4.4.1. Análisis de características visuales con PDI

En esta etapa se realizó el análisis de características visuales mediante procesamiento digital de imágenes, con la finalidad de observar las diferencias presentes entre un billete verdadero y un billete falsificado. Para ello, se aplicaron técnicas de transformación y extracción de rasgos visuales, considerando la conversión a escala de grises, la detección de bordes mediante Canny, el operador Sobel, el filtro Laplaciano y el análisis frecuencial mediante la Transformada Rápida de Fourier. Este procedimiento permitió examinar la forma, el contraste, los contornos, la textura y la distribución de frecuencias presentes en cada imagen.

El procesamiento inició con la imagen original del billete, representada como una matriz digital de píxeles. Una imagen en color puede expresarse de la siguiente manera:

$$I(x, y) = \{R(x, y), G(x, y), B(x, y)\}$$

Donde:

$I(x, y)$: imagen digital en la posición espacial (x, y).

$R(x, y)$: componente roja del píxel.

$G(x, y)$: componente verde del píxel.

$B(x, y)$: componente azul del píxel.

Posteriormente, la imagen fue convertida a escala de grises para reducir la información cromática y conservar la intensidad luminosa. Esta conversión se representa mediante la siguiente expresión:

$$I_g(x,y) = 0.299R(x,y) + 0.587G(x,y) + 0.114B(x,y)$$

Donde:

$I_g(x,y)$: intensidad de la imagen en escala de grises.

$R(x,y)$, $G(x,y)$, $B(x,y)$: valores de los canales rojo, verde y azul.

Esta transformación permitió trabajar con una sola matriz de intensidad, facilitando la aplicación de filtros posteriores y reduciendo la complejidad del procesamiento.

En la Figura 34 se muestra la comparación de técnicas de procesamiento digital aplicadas a un billete verdadero y a un billete falso.

- En la primera columna se presentan las imágenes originales. En el billete verdadero se observa una estructura visual más definida, con mejor conservación de contornos, tonalidades y detalles impresos. En cambio, el billete falso presenta menor nitidez, ruido visual y pérdida de uniformidad, lo que dificulta su identificación mediante observación directa.
- La segunda columna corresponde a la conversión en escala de grises. En el billete verdadero se mantiene una mejor separación entre zonas claras y oscuras, lo que permite distinguir con mayor precisión los elementos gráficos principales. En el billete falso, la imagen en grises muestra pérdida de detalle, iluminación irregular y menor contraste, lo que afecta la lectura visual de sus características.
- La tercera columna presenta la detección de bordes mediante el método Canny. Esta técnica permite resaltar los cambios bruscos de intensidad y delimitar los contornos principales del billete. En el caso del billete verdadero, los bordes aparecen con mayor continuidad y definición. Por el contrario, en el billete falso se observan bordes fragmentados, zonas discontinuas y presencia de puntos aislados, asociados al ruido y a la baja calidad visual de la imagen.
- La cuarta columna muestra el resultado del operador Sobel, el cual permite detectar variaciones de intensidad en las direcciones horizontal y vertical. En el billete verdadero, los gradientes se concentran alrededor de los elementos gráficos y bordes

principales. En el billete falso, la respuesta del operador se presenta más dispersa, con menor continuidad en los contornos y mayor presencia de ruido.

- La quinta columna corresponde al filtro Laplaciano, utilizado para resaltar cambios rápidos de intensidad y detalles finos. En el billete verdadero, este filtro permite reconocer la silueta general del billete y ciertos rasgos internos del diseño. En el billete falso, la respuesta aparece más afectada por patrones irregulares y ruido, reduciendo la claridad de los detalles relevantes.

El análisis frecuencial mediante la Transformada Rápida de Fourier permite estudiar la distribución de frecuencias de la imagen. De forma general, la transformada bidimensional se expresa como:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x,y)\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Donde:

$F(u,v)$: representación de la imagen en el dominio de la frecuencia.

$I(x,y)$: intensidad del píxel en la posición espacial (x, y).

M, N: dimensiones de la imagen.

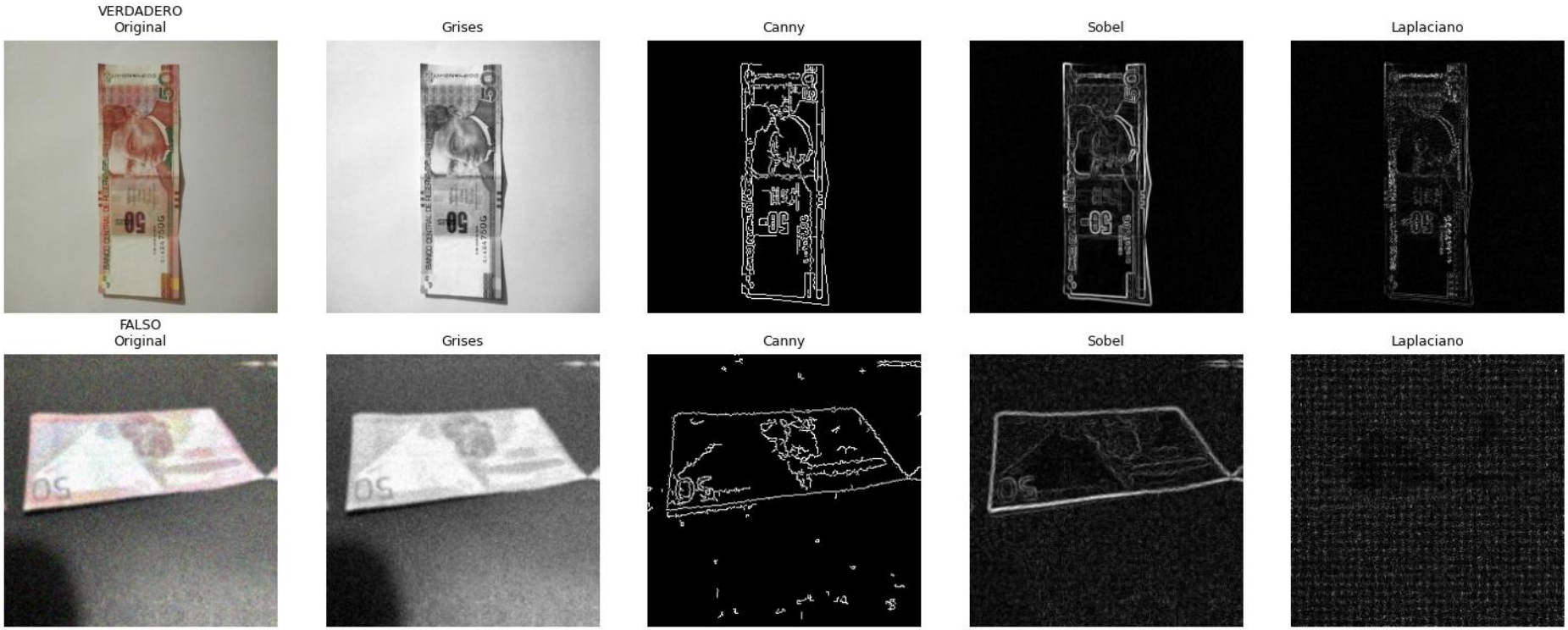
u, v: coordenadas en el dominio frecuencial.

j: unidad imaginaria.

Esta representación permite observar la concentración de energía en bajas y altas frecuencias. En el billete verdadero, la distribución frecuencial tiende a ser más estable alrededor del centro del espectro, debido a la presencia de patrones visuales definidos. En el billete falso, se aprecian alteraciones relacionadas con ruido, textura irregular y pérdida de información estructural.

Figura 34

Comparación de técnicas de procesamiento digital para la detección de billetes verdaderos y falsos



Nota. Elaboración propia

4.4.2. Generadores de datos (aumentación solo en train)

En esta sección se describe la configuración de los generadores de datos empleados para el entrenamiento, validación y prueba de los modelos de clasificación de billetes peruanos. El procedimiento se desarrolló considerando dos clases: verdadero y falso, donde la clase falso fue definida como índice positivo. Esta organización permitió estructurar el problema como una clasificación binaria, coherente con el propósito de identificar si una imagen corresponde a un billete auténtico o falsificado. Este apartado se vincula con la arquitectura experimental planteada en el documento, donde se comparan modelos CNN, CNN-LSTM y CNN-GRU aplicados a visión por computadora para la identificación de billetes falsificados.

La aumentación de datos se aplicó únicamente al conjunto de entrenamiento, con el fin de incrementar la variabilidad visual de las imágenes sin modificar las imágenes utilizadas en validación y prueba. Esta decisión permitió que el modelo aprenda a reconocer billetes bajo diferentes condiciones de rotación, escala, desplazamiento, brillo y orientación, manteniendo una evaluación más objetiva sobre imágenes no alteradas artificialmente.

El proceso de aumentación puede representarse como una transformación aplicada sobre la imagen original:

$$I_a = T(I)$$

Donde:

I_a: imagen aumentada generada para entrenamiento.

I: imagen original del conjunto de datos.

T: conjunto de transformaciones aplicadas a la imagen.

En este caso, la transformación T estuvo conformada por rotación, zoom, desplazamiento horizontal, desplazamiento vertical, variación de brillo y volteo horizontal:

$$T(I) = \{R_\theta, Z_s, D_x, D_y, B_\alpha, H_f\}$$

Donde:

R_θ: rotación aplicada a la imagen.

Z_s: variación de escala o zoom.

D_x: desplazamiento horizontal.

D_y: desplazamiento vertical.

B_α: ajuste de brillo.

H_f: volteo horizontal.

En la Figura 35 se muestra la configuración del generador de imágenes mediante ImageDataGenerator. Para el conjunto de entrenamiento se emplearon transformaciones como rotation_range=12, zoom_range=0.12, width_shift_range=0.08, height_shift_range=0.08, brightness_range=(0.8,1.2) y horizontal_flip=True. Estos parámetros permitieron generar variaciones controladas de las imágenes, simulando condiciones de captura más cercanas a situaciones reales, como inclinación del billete, cambios de iluminación, desplazamientos dentro del encuadre o ligeras deformaciones visuales.

Figura 35

Aumento de datos para entrenamiento del modelo

```
CLASES = ["verdadero", "falso"]   # falso -> indice 1 (positivo)

train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=12, zoom_range=0.12, width_shift_range=0.08,
    height_shift_range=0.08, brightness_range=(0.8,1.2), horizontal_flip=True)
eval_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

def gen(df_, datagen, shuffle):
    return datagen.flow_from_dataframe(
        df_, x_col="filepath", y_col="label", classes=CLASES,
        target_size=(IMG_SIZE,IMG_SIZE), class_mode="binary",
        batch_size=BATCH_SIZE, shuffle=shuffle, seed=SEED)

train_gen = gen(train_df, train_datagen, True)
val_gen   = gen(val_df,   eval_datagen, False)
test_gen  = gen(test_df,  eval_datagen, False)
print("Mapa de clases:", train_gen.class_indices)

# Muestra de un batch aumentado
xb, yb = next(train_gen)
fig, axes = plt.subplots(2, 6, figsize=(16,5))
for i, ax in enumerate(axes.flat):
    im = (xb[i]-xb[i].min())/(xb[i].max()-xb[i].min()+1e-6)
    ax.imshow(im); ax.axis("off")
    ax.set_title("FALSO" if yb[i]==1 else "VERDADERO", fontsize=9,
```

Nota. Elaboración propia

La rotación se puede expresar de forma matricial mediante:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Donde:

x, y: coordenadas originales del píxel.

x', y': coordenadas transformadas luego de la rotación.

θ: ángulo de rotación aplicado.

Asimismo, el desplazamiento horizontal y vertical puede expresarse como:

$$x' = x + \Delta x$$

$$y' = y + \Delta y$$

Donde:

Δx: variación aplicada sobre el eje horizontal.

Δy: variación aplicada sobre el eje vertical.

Estas operaciones contribuyeron a que el modelo no dependa estrictamente de una única posición del billete dentro de la imagen. Por ello, si el billete aparece ligeramente desplazado, inclinado o con diferente tamaño, la red puede conservar capacidad de reconocimiento sobre sus patrones visuales.

Para los conjuntos de validación y prueba no se aplicó aumentación de datos. En estos casos, el generador solo ejecutó la función de preprocesamiento correspondiente al modelo utilizado. Esta separación evitó que las métricas de evaluación fueran afectadas por imágenes modificadas artificialmente. Por tanto, la validación y la prueba reflejaron el comportamiento del modelo sobre imágenes conservadas en condiciones más estables. La clasificación binaria fue definida mediante el parámetro class_mode=''binary", considerando las clases:

$$C = \{\text{verdadero,falso}\}$$

La etiqueta de salida puede representarse como:

$$y = \begin{cases} 0, & \text{si el billete es verdadero} \\ 1, & \text{si el billete es falso} \end{cases}$$

Esta codificación permitió que el modelo entregue una probabilidad asociada a la clase positiva. De manera general, la salida del clasificador puede expresarse como:

$$\hat{y} = P(y = 1 \mid I)$$

Donde:

$\hat{y}$: probabilidad estimada por el modelo.

$P(y = 1 \mid I)$: probabilidad de que la imagen Icorresponda a un billete falso.

I: imagen de entrada procesada por la red.

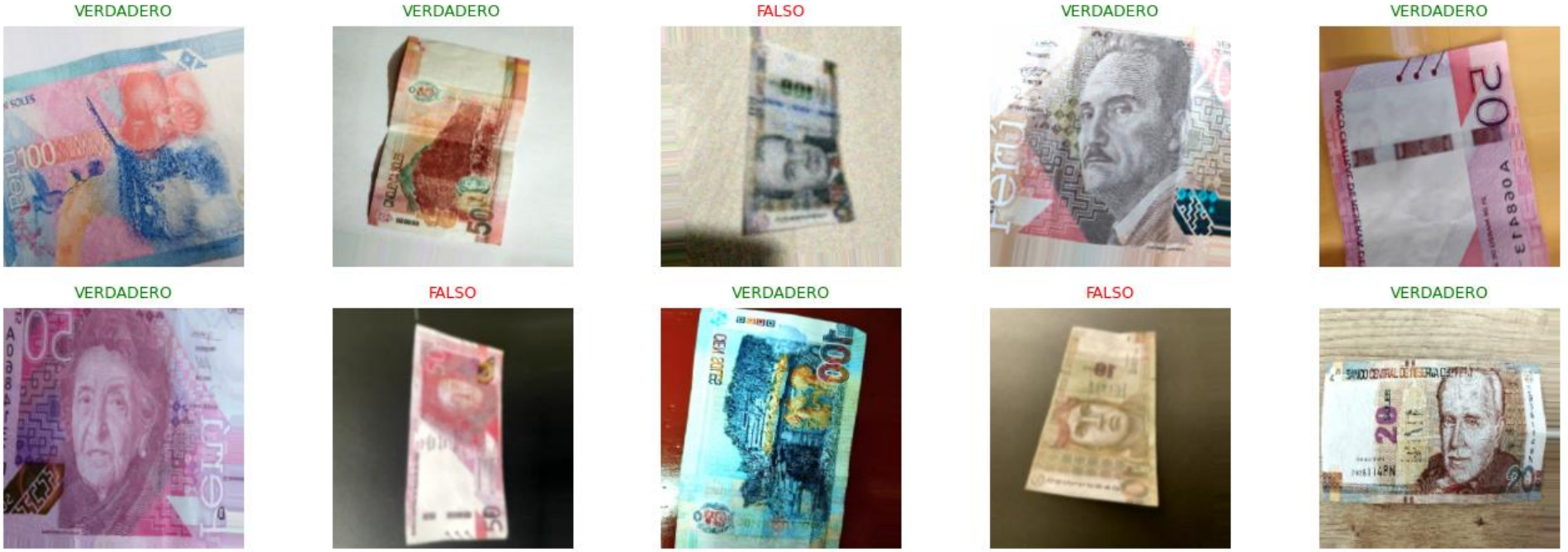
En la parte final de la Figura 35 se observa la extracción de un lote aumentado mediante next(train_gen). Este procedimiento permitió visualizar un conjunto de imágenes generadas por el flujo de entrenamiento y verificar que las etiquetas asignadas correspondieran correctamente a las clases verdadero o falso. Además, se normalizó cada imagen del lote para facilitar su visualización, ajustando los valores de intensidad dentro de un rango adecuado.

La Figura 36 presenta una muestra de imágenes clasificadas como verdaderas o falsas. En esta visualización se observan diferentes denominaciones, ángulos de captura, niveles de iluminación, fondos y grados de nitidez. Las imágenes rotuladas como VERDADERO representan billetes auténticos presentes en el conjunto de datos, mientras que las imágenes rotuladas como FALSO corresponden a billetes falsificados. Esta muestra permite evidenciar la variabilidad visual que enfrenta el modelo durante el proceso de aprendizaje y evaluación.

En las imágenes verdaderas se aprecia mayor presencia de detalles gráficos, texturas más definidas y elementos visuales mejor conservados. En cambio, algunas imágenes falsas presentan desenfoque, baja calidad de impresión, pérdida de contraste o distorsiones en la forma del billete. Estas diferencias no siempre son evidentes a simple vista; por ello, el uso de redes neuronales convolucionales resulta adecuado para extraer patrones espaciales de mayor profundidad.

Figura **36**

Muestra de imágenes clasificadas por el modelo verdadero o falso



Nota. Elaboración propia

4.4.3. Entrenamiento de los tres modelos CNN, CNN-LSTM y CNN-GRU

El entrenamiento de los modelos se desarrolló con tres arquitecturas de aprendizaje profundo: CNN, CNN-LSTM y CNN-GRU. La finalidad fue comparar el comportamiento de cada red durante el aprendizaje de patrones visuales asociados a billetes verdaderos y falsificados. Para ello, se emplearon imágenes previamente organizadas en dos clases, considerando la clase falso como clase positiva del modelo.

El proceso de entrenamiento se realizó por épocas, donde cada época corresponde a una pasada completa del conjunto de datos de entrenamiento por la red neuronal. En cada iteración, el modelo ajustó sus pesos internos mediante la reducción progresiva de la función de pérdida. De manera general, el aprendizaje supervisado puede expresarse como:

$$\hat{y} = f(x; \theta)$$

Donde:

x: imagen de entrada del billete.

$\hat{y}$: salida estimada por el modelo.

f: función aprendida por la red neuronal.

θ: conjunto de pesos y sesgos ajustados durante el entrenamiento.

La salida del modelo se comparó con la etiqueta real mediante una función de pérdida. Al tratarse de una clasificación binaria, se utilizó la entropía cruzada binaria, expresada como:

$$L(y, \hat{y}) = -[y\log(\hat{y}) + (1 - y)\log(1 - \hat{y})]$$

Donde:

$L(y, \hat{y})$: pérdida calculada entre la etiqueta real y la predicción.

y: etiqueta real de la imagen, donde 0 representa verdadero y 1 representa falso.

$\hat{y}$: probabilidad estimada por el modelo para la clase positiva.

El objetivo del entrenamiento fue minimizar dicha pérdida mediante la actualización de parámetros:

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L}{\partial \theta}$$

Donde:

θ_{t+1}: parámetros actualizados del modelo.

θ_t: parámetros en la iteración actual.

η: tasa de aprendizaje.

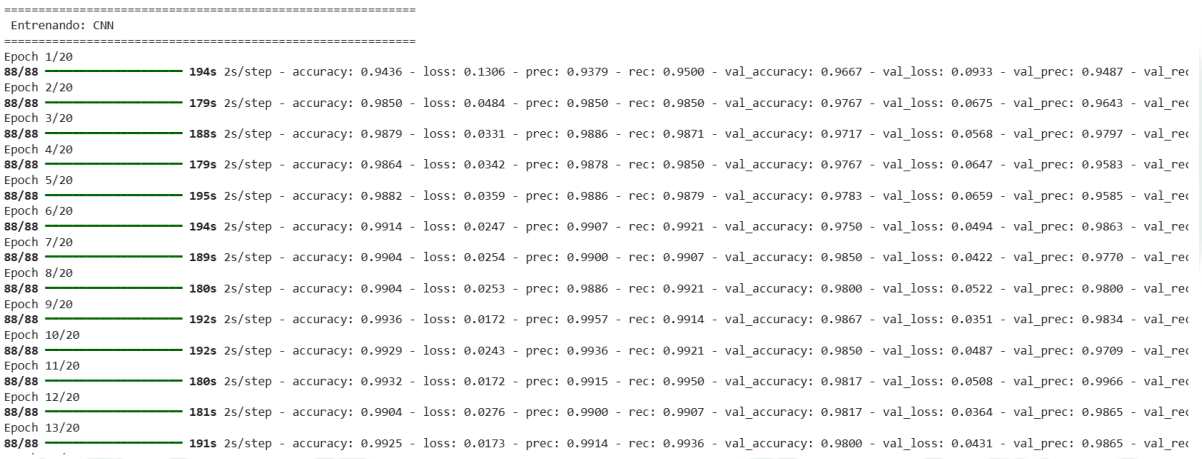
$\frac{\partial L}{\partial \theta}$: gradiente de la pérdida respecto a los parámetros.

Este procedimiento permitió que cada modelo ajustara progresivamente sus filtros, pesos y conexiones internas para mejorar la identificación de billetes falsificados.

En la Figura 37 se muestra el proceso de entrenamiento del modelo CNN. Esta arquitectura se encargó de extraer características espaciales directamente desde las imágenes de entrada, tales como bordes, texturas, zonas de contraste, detalles impresos y patrones visuales asociados a los billetes.

Figura 37

Proceso de entrenamiento del modelo CNN



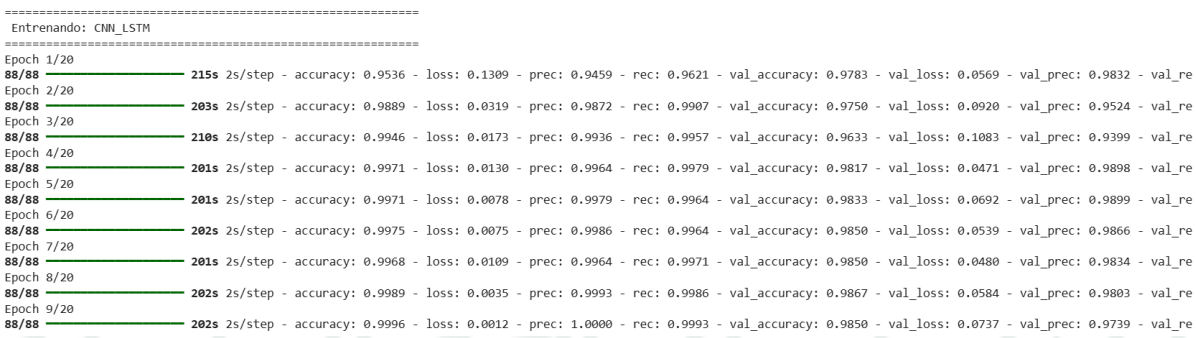
Nota. Elaboración propia

Durante las primeras épocas, el modelo inició con una exactitud de entrenamiento de 0.9436 y una pérdida de 0.1306. Conforme avanzó el entrenamiento, la exactitud aumentó hasta valores cercanos a 0.99, mientras que la pérdida disminuyó progresivamente hasta valores próximos a 0.0173. Este comportamiento indica que la CNN logró aprender patrones visuales diferenciables entre billetes verdaderos y falsos. En validación, el modelo mantuvo una exactitud elevada, con valores aproximados entre 0.9667 y 0.9867 durante las épocas observadas. Asimismo, la pérdida de validación presentó una reducción desde 0.0933 hasta valores cercanos a 0.0431, lo cual evidencia una respuesta estable frente a datos no utilizados directamente para el ajuste de pesos.

La Figura 38 presenta el entrenamiento del modelo CNN-LSTM. Esta arquitectura combinó la extracción de características espaciales mediante capas convolucionales con el procesamiento secuencial de una red LSTM. La intención fue complementar la información visual de la CNN con una estructura capaz de retener dependencias internas dentro de las características extraídas.

Figura 38

Entrenamiento del modelo CNN-LSTM



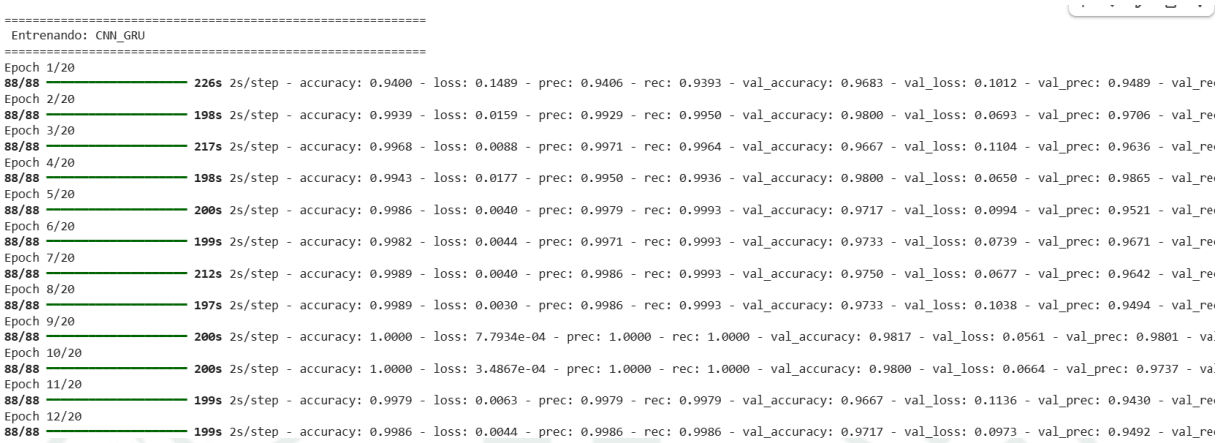
Nota. Elaboración propia

En las primeras épocas, el modelo registró una exactitud de entrenamiento de 0.9536 y una pérdida de 0.1309. Luego, la exactitud aumentó rápidamente hasta valores superiores a 0.99, mientras que la pérdida descendió hasta aproximadamente 0.0012 en las últimas épocas observadas. Este comportamiento muestra que el modelo logró ajustarse con rapidez al conjunto de entrenamiento. Sin embargo, en validación se observó una mayor variación de la pérdida en comparación con la CNN. Aunque la exactitud de validación se mantuvo en valores altos, entre aproximadamente 0.9633 y 0.9867, la pérdida no siguió una reducción completamente uniforme. Este comportamiento sugiere que el modelo aprendió adecuadamente, pero con ciertas oscilaciones en su capacidad de generalización.

La Figura 39 muestra el entrenamiento del modelo CNN-GRU. Esta arquitectura también combinó una etapa convolucional con una red recurrente, pero utilizando unidades GRU, las cuales presentan una estructura más compacta que las LSTM. Su función fue procesar las características extraídas por la CNN y evaluar si una memoria recurrente más ligera podía mejorar la clasificación.

Figura 39

Entrenamiento del modelo CNN-GRU



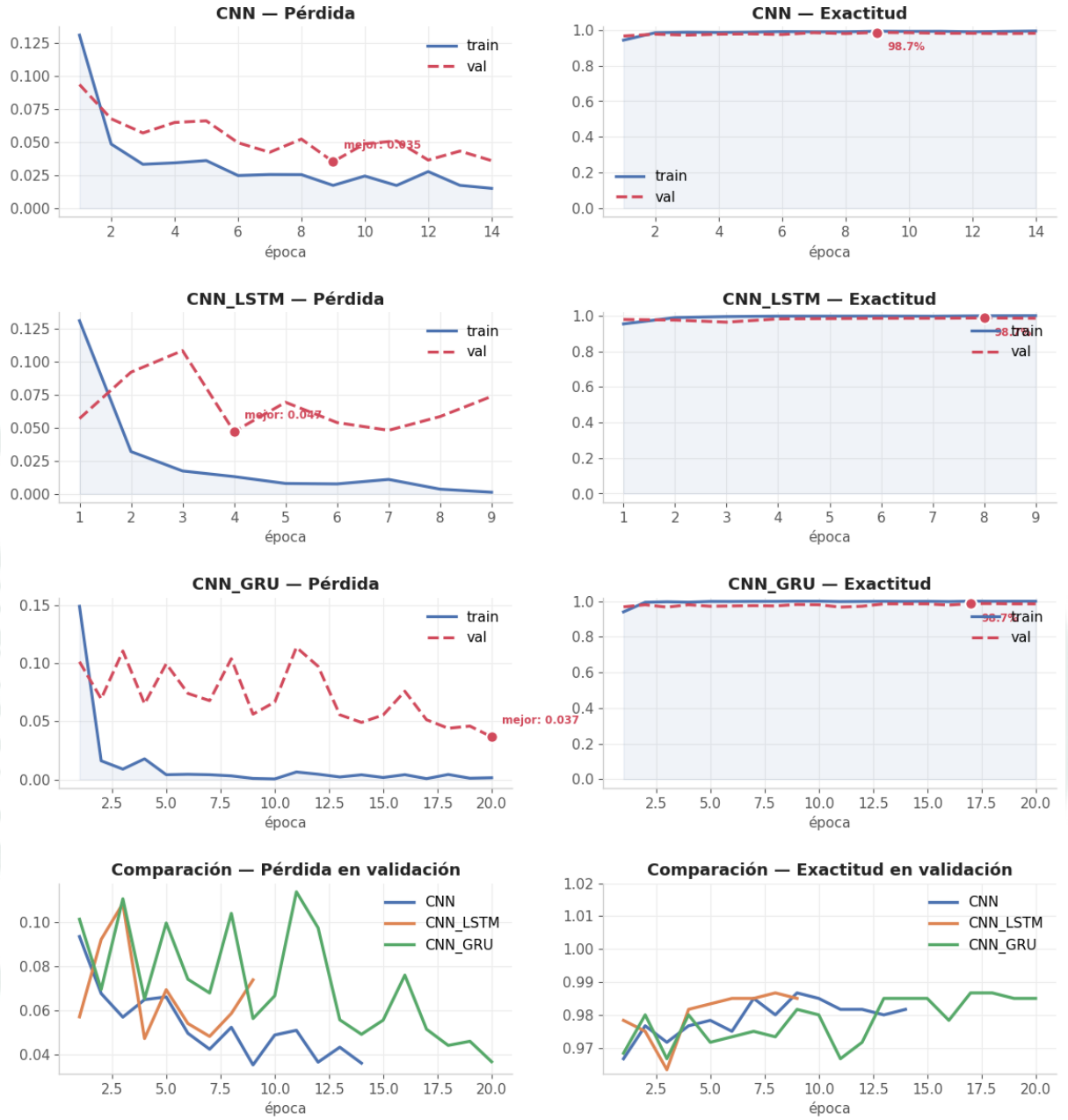
Nota. Elaboración propia

Durante el entrenamiento, el modelo inició con una exactitud de 0.9400 y una pérdida de 0.1489. A partir de las siguientes épocas, la exactitud aumentó de manera rápida hasta valores cercanos a 1.0000, mientras que la pérdida se redujo hasta niveles muy bajos, incluso menores a 0.0010 en algunas épocas. Este descenso evidencia un ajuste intenso del modelo sobre el conjunto de entrenamiento. En validación, la exactitud se mantuvo generalmente alta, con valores cercanos a 0.9667, 0.9800 y 0.9817 según las épocas observadas. Sin embargo, la pérdida de validación presentó oscilaciones, alcanzando valores como 0.1104, 0.1038 y 0.1136 en algunos puntos. Esto indica que, aunque el modelo consiguió un alto ajuste en entrenamiento, su comportamiento en validación tuvo variaciones más marcadas que en la CNN.

La Figura 40 presenta las curvas de entrenamiento y validación de los tres modelos evaluados. En la primera columna se observan las curvas de pérdida, mientras que en la segunda columna se muestran las curvas de exactitud. Además, en la parte inferior se presenta una comparación directa de la pérdida y exactitud de validación entre CNN, CNN-LSTM y CNN-GRU.

Figura 40

Curvas de entrenamiento y validación de los modelos CNN, CNN-LSTM y CNN-GRU



Nota. Elaboración propia

- En el modelo CNN, la pérdida de entrenamiento disminuyó de forma progresiva y la pérdida de validación se mantuvo relativamente estable, con una mejor pérdida aproximada de 0.0935 según la anotación de la figura. La exactitud de validación se mantuvo cercana a 98. %, lo que evidencia una convergencia adecuada y una diferencia reducida entre entrenamiento y validación.

- En el modelo CNN-LSTM, la pérdida de entrenamiento descendió de forma rápida; sin embargo, la pérdida de validación mostró una variación inicial más elevada, con una mejora aproximada de 0.0474. La exactitud de validación se mantuvo alrededor de 98%, con un comportamiento estable después de las primeras épocas.
- En el modelo CNN-GRU, la pérdida de entrenamiento descendió con mayor rapidez, llegando a valores cercanos a cero. Sin embargo, la pérdida de validación presentó fluctuaciones más notorias a lo largo de las épocas, aunque alcanzó una mejor pérdida aproximada de 0.037. La exactitud de validación se mantuvo cercana a 98.7%, mostrando un rendimiento alto, aunque con mayor variabilidad en la curva de pérdida.

4.4.4. Evaluación en el conjunto de Test

La evaluación final de los modelos se realizó utilizando el conjunto de prueba, el cual no intervino en el proceso de entrenamiento ni en la validación. Esta etapa permitió medir el desempeño real de las arquitecturas CNN, CNN-LSTM y CNN-GRU frente a imágenes no utilizadas durante el ajuste de los pesos. Para este análisis se consideraron las métricas de exactitud, precisión, recall, F1-score, matriz de confusión y AUC, debido a que permiten observar no solo el porcentaje global de aciertos, sino también el comportamiento específico en la detección de billetes falsificados.

El conjunto de prueba estuvo conformado por 561 imágenes, distribuidas en 261 billetes verdaderos y 300 billetes falsos. La clase falso fue considerada como la clase positiva, debido a que representa el evento de mayor interés dentro de la investigación. Por ello, el análisis se centró especialmente en la capacidad de cada modelo para reconocer billetes falsificados sin confundirlos con billetes verdaderos. La matriz de confusión permitió organizar los aciertos y errores de clasificación de cada modelo. De forma general, esta matriz se estructura mediante cuatro componentes:

$$CM = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

Donde:

TN: billetes verdaderos clasificados correctamente como verdaderos.

FP: billetes verdaderos clasificados incorrectamente como falsos.

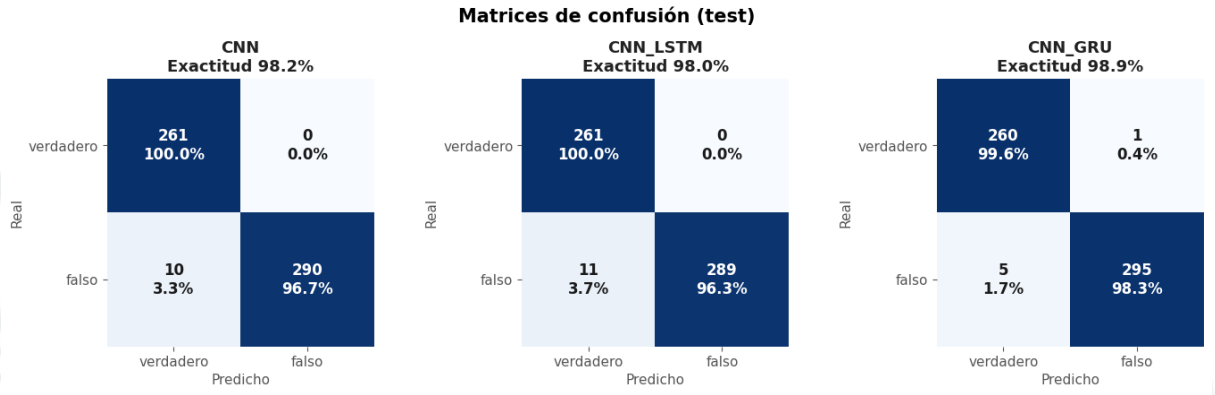
FN: billetes falsos clasificados incorrectamente como verdaderos.

TP: billetes falsos clasificados correctamente como falsos.

En la Figura 41 se muestran las matrices de confusión obtenidas para los tres modelos evaluados. En el caso del modelo CNN, se clasificaron correctamente los 261 billetes verdaderos, sin registrar falsos positivos. Asimismo, de los 300 billetes falsos, 290 fueron clasificados correctamente y 10 fueron confundidos como verdaderos. Este resultado generó una exactitud de 98.2%, con una detección de billetes falsos de 96.7%.

Figura 41

Matrices de confusión de los modelos evaluados



Nota. Elaboración propia

Para el modelo CNN-LSTM, también se clasificaron correctamente los 261 billetes verdaderos, sin presencia de falsos positivos. Sin embargo, de los 300 billetes falsos, 289 fueron identificados correctamente y 11 fueron clasificados de manera errónea como verdaderos. Este comportamiento produjo una exactitud de 98.0 %, ligeramente menor que la obtenida por la CNN.

En el modelo CNN-GRU se obtuvo el mejor comportamiento general. Este modelo clasificó correctamente 260 billetes verdaderos y presentó solo 1 falso positivo. Respecto a los billetes falsos, identificó correctamente 295 de 300, registrando únicamente 5 falsos negativos. Este resultado permitió alcanzar una exactitud de 98.93%, siendo el valor más alto entre los tres modelos evaluados.

La exactitud global se calculó mediante la siguiente expresión:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Para el modelo CNN-GRU, considerando los valores de la matriz de confusión:

$$Accuracy = \frac{295 + 260}{295 + 260 + 1 + 5}$$

$$Accuracy = \frac{555}{561} = 0.9893$$
$$Accuracy = 98.93\%$$

Este resultado indica que el modelo CNN-GRU clasificó correctamente 555 imágenes de las 561 evaluadas, presentando solo 6 errores en el conjunto de prueba.

La precisión para la clase falso se calculó mediante:

$$Precision = \frac{TP}{TP + FP}$$

Para CNN-GRU:

$$Precision = \frac{295}{295 + 1}$$
$$Precision = \frac{295}{296} = 0.9966$$
$$Precision = 99.66\%$$

Este valor indica que, cuando el modelo CNN-GRU clasificó una imagen como falsa, en el 99.66% de los casos dicha predicción fue correcta.

El recall para la clase falso se calculó mediante:

$$Recall = \frac{TP}{TP + FN}$$

Para CNN-GRU:

$$Recall = \frac{295}{295 + 5}$$
$$Recall = \frac{295}{300} = 0.9833$$
$$Recall = 98.33\%$$

Este resultado muestra que el modelo CNN-GRU logró detectar el 98.33 % de los billetes falsos presentes en el conjunto de prueba.

El F1-score se obtuvo a partir de la relación entre precisión y recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Para CNN-GRU:

$$F1 = 2 \cdot \frac{0.9966 \cdot 0.9833}{0.9966 + 0.9833}$$

$$F1 = 0.9899$$

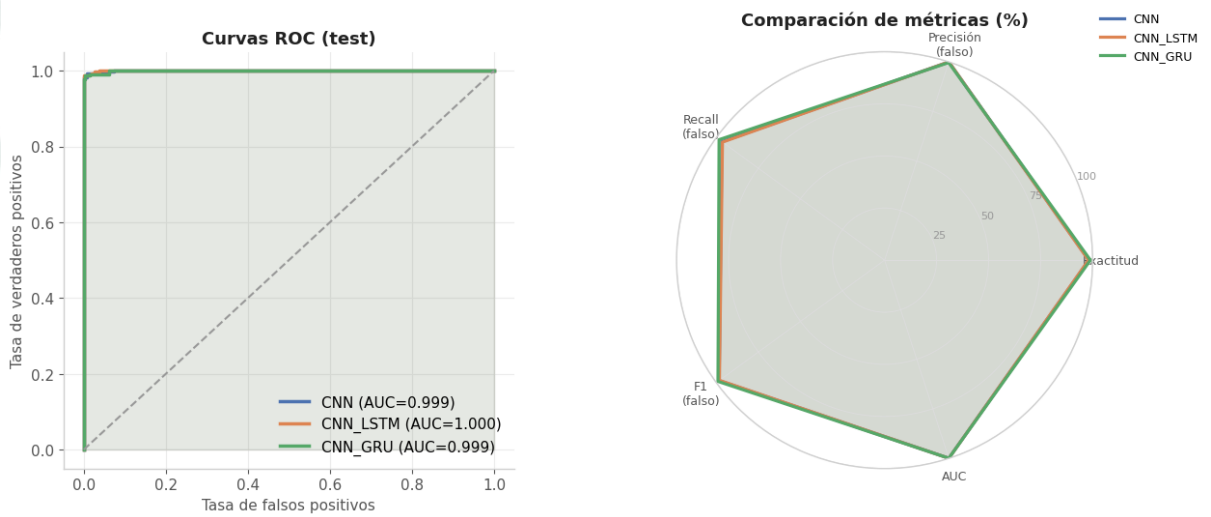
$$F1 = 98.99\%$$

Este valor confirma que el modelo CNN-GRU mantuvo un equilibrio más adecuado entre la precisión y la sensibilidad para la clase falso.

En la Figura 42 se presentan las curvas ROC y la comparación de métricas de los modelos evaluados. Las curvas ROC se ubicaron cerca del extremo superior izquierdo, lo cual indica una alta capacidad de discriminación entre billetes verdaderos y falsificados. El modelo CNN obtuvo un AUC de 0.9995, el modelo CNN-LSTM alcanzó un AUC de 0.9997 y el modelo CNN-GRU registró un AUC de 0.9993. Aunque los tres valores son elevados, el AUC del CNN-LSTM fue ligeramente mayor. No obstante, al considerar de forma conjunta la exactitud, el recall y el F1-score para la clase falso, el modelo CNN-GRU presentó mejor desempeño práctico.

Figura 42

Curvas ROC y comparación de métricas de los modelos evaluados



Nota. Elaboración propia

La Tabla 7 muestra el reporte detallado por modelo en el conjunto de prueba. En el modelo CNN, la clase verdadero obtuvo una precisión de 0.963, recall de 1.000 y F1-score de

0.981. Para la clase falso, obtuvo precisión de 1.000, recall de 0.967 y F1-score de 0.983. Estos resultados muestran que el modelo CNN no confundió billetes verdaderos como falsos, pero dejó sin detectar 10 billetes falsificados.

Tabla 7

Desempeño por modelo en prueba

Modelo	Clase	Precisión	Recall	F1-score	Support
CNN	Verdadero	0.963	1.000	0.981	261
CNN	Falso	1.000	0.967	0.983	300
CNN	Accuracy	---	---	0.982	561
CNN	Macro avg	0.982	0.983	0.982	561
CNN	Weighted avg	0.983	0.982	0.982	561
CNN_LSTM	Verdadero	0.960	1.000	0.979	261
CNN_LSTM	Falso	1.000	0.963	0.981	300
CNN_LSTM	Accuracy	---	---	0.980	561
CNN_LSTM	Macro avg	0.980	0.982	0.980	561
CNN_LSTM	Weighted avg	0.981	0.980	0.980	561
CNN_GRU	Verdadero	0.981	0.996	0.989	261
CNN_GRU	Falso	0.997	0.983	0.990	300
CNN_GRU	Accuracy	---	---	0.989	561
CNN_GRU	Macro avg	0.989	0.990	0.989	561
CNN_GRU	Weighted avg	0.989	0.989	0.989	561

Nota. Elaboración propia

En el caso del modelo CNN-LSTM, la clase verdadero obtuvo precisión de 0.960, recall de 1.000 y F1-score de 0.979. Para la clase falso, obtuvo precisión de 1.000, recall de 0.963 y F1-score de 0.981. Este modelo mantuvo una precisión perfecta para la clase falso, pero presentó el menor recall entre los tres modelos, debido a que clasificó incorrectamente 11 billetes falsos como verdaderos. Por su parte, el modelo CNN-GRU alcanzó una precisión de 0.981, recall de 0.996 y F1-score de 0.989 para la clase verdadero. Para la clase falso, registró una precisión de 0.997, recall de 0.983 y F1-score de 0.990. Estos resultados muestran un comportamiento más equilibrado, debido a que redujo los falsos negativos y mantuvo una cantidad mínima de falsos positivos.

La Tabla 8 resume la comparación general de los tres modelos. La CNN obtuvo una exactitud de 98.22 %, precisión falso de 100.00%, recall falso de 96.67%, F1 falso de 98.3% y AUC de 0.9995. La CNN-LSTM presentó una exactitud de 98.04%, precisión falso de 100.00 %, recall falso de 96.33%, F1 falso de 98.13% y AUC de 0.9997. Finalmente, la CNN-GRU

alcanzó una exactitud de 98.93%, precisión falso de 99.66 %, recall falso de 98.33%, F1 falso de 98.99% y AUC de 0.9993.

Tabla 8

Comparación general de modelos en prueba

Modelo	Exactitud (%)	Precisión falso (%)	Recall falso (%)	F1 falso (%)	AUC
CNN	98.22	100.00	96.67	98.31	0.9995
CNN_LSTM	98.04	100.00	96.33	98.13	0.9997
CNN_GRU	98.93	99.66	98.33	98.99	0.9993

Nota. Elaboración propia

A partir de estos resultados, se observa que el modelo CNN-GRU presentó el mejor desempeño global en el conjunto de prueba. Aunque su precisión para la clase falso fue ligeramente menor que la de CNN y CNN-LSTM, logró un mayor recall y un mayor F1-score. Esto resulta relevante en el contexto de la investigación, debido a que la identificación de billetes falsificados requiere reducir la cantidad de falsos negativos, es decir, evitar que billetes falsos sean clasificados como verdaderos.

CAPÍTULO V

RESULTADOS Y ANÁLISIS

5.1. Evaluación de modelos CNN y RNN

La evaluación de los modelos CNN, CNN-LSTM y CNN-GRU se realizó con el propósito de comprobar su comportamiento frente a imágenes de billetes sometidas a distintas condiciones de captura. En esta etapa no solo se consideró la clasificación general entre billete verdadero y falso, sino también la estabilidad de la predicción cuando la imagen presenta variaciones de iluminación, exposición, desenfoque, ruido, rotación y baja resolución. Este procedimiento permitió analizar la capacidad de cada arquitectura para mantener una respuesta coherente ante alteraciones visuales que pueden presentarse en escenarios reales de uso.

En el contexto de la investigación, la CNN se orientó a la extracción de características espaciales de la imagen, mientras que las arquitecturas CNN-LSTM y CNN-GRU incorporaron una etapa recurrente para procesar la representación obtenida por la red convolucional. De esta manera, se buscó comparar si la incorporación de estructuras recurrentes mejora la identificación de billetes peruanos falsificados frente a una red CNN convencional, en concordancia con el enfoque metodológico planteado en el documento de tesis.

La predicción de cada modelo se realizó a partir de una probabilidad asociada a la clase positiva, definida como billete falso. Matemáticamente, la salida puede expresarse como:

$$\hat{y} = P(y = 1 \mid I)$$

Donde:

$\hat{y}$: probabilidad estimada por el modelo.

$P(y = 1 \mid I)$: probabilidad de que la imagen Icorresponda a un billete falso.

I: imagen de entrada procesada por el modelo.

$y = 1$: clase falso.

La decisión final se estableció mediante un umbral de clasificación de 0.5:

$$Clase = \begin{cases} FALSO, & \text{si } \hat{y} \geq 0.5 \\ VERDADERO, & \text{si } \hat{y} < 0.5 \end{cases}$$

Figura 43

Función de predicción de billetes mediante el mejor modelo entrenado

```
def predecir_billete(path, model, nombre_modelo="modelo"):
    img = cv2.imread(path)
    if img is None:
        print("No se pudo leer:", path); return None
    rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    x = cv2.resize(rgb, (IMG_SIZE, IMG_SIZE)).astype("float32")
    x = preprocess_input(np.expand_dims(x.copy(), 0))
    p = float(model.predict(x, verbose=0).ravel()[0])      # P(falso)
    etiqueta = "FALSO" if p >= 0.5 else "VERDADERO"
    conf = p if p >= 0.5 else 1 - p
    plt.figure(figsize=(4,4)); plt.imshow(rgb); plt.axis("off")
    plt.title(f"{etiqueta}  ({conf*100:.1f}%)\n[{nombre_modelo}]",
              color=("red" if etiqueta=="FALSO" else "green"), fontsize=13, fontweight="bold")
    plt.show()
    return etiqueta, conf

modelo_final = modelos[mejor]
print(f"Usando el mejor modelo: {mejor}. Sube imágenes de billetes para probar:")
from google.colab import files
subidas = files.upload()
for nombre in subidas:
    predecir_billete(nombre, modelo_final, mejor)
```

Nota. Elaboración propia

Este criterio permitió convertir la probabilidad generada por la red en una etiqueta binaria. Por tanto, si el valor entregado por el modelo fue igual o superior a 0.5, la imagen se clasificó como falsa; en caso contrario, se clasificó como verdadera.

5.1.1. Prueba para un billete de 100 soles – enfoque verdadero

Para la primera prueba se utilizó una imagen correspondiente a un billete verdadero de 100 soles. El objetivo fue evaluar si los modelos entrenados podían reconocer correctamente un billete auténtico, incluso cuando la imagen era sometida a distintas alteraciones visuales. Esta prueba permitió analizar la robustez de los modelos frente a condiciones de captura no ideales, como baja iluminación, sobreexposición, desenfoque, ruido, rotación y reducción de resolución.

En la Figura 44 se muestra el billete verdadero de 100 soles bajo siete condiciones de captura: original, baja luz, sobreexposición, desenfoque, ruido, rotación y baja resolución. La imagen original presenta una estructura visual definida, con elementos gráficos reconocibles y una adecuada distribución de color. En la condición de baja luz, la imagen reduce su intensidad,

pero conserva rasgos suficientes para su reconocimiento. En la imagen sobreexpuesta, algunas zonas pierden detalle por exceso de iluminación, aunque todavía se mantiene la forma general del billete.

Figura 44

Reconocimiento de captura de un Billete verdadero de 100 soles



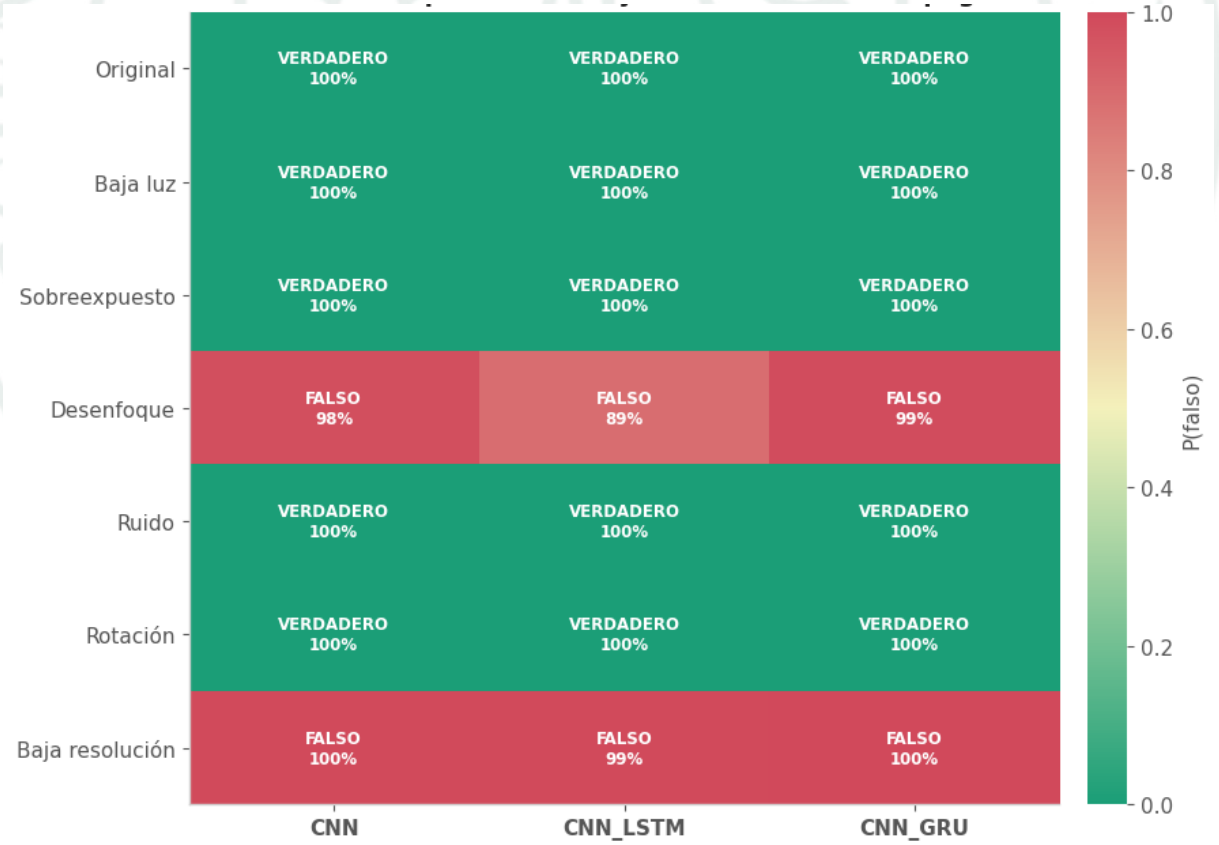


Nota. Elaboración propia

La Figura 45 presenta la predicción obtenida por cada modelo bajo las distintas condiciones evaluadas.

Figura 45

Predicción por condición y modelo de un Billete verdadero de 100 soles



Nota. Elaboración propia

En las condiciones original, baja luz, sobreexposición, ruido y rotación, los tres modelos clasificaron el billete como verdadero con una confianza de 100%. Este resultado indica que, ante variaciones moderadas de iluminación, exposición, ruido y orientación, las arquitecturas mantuvieron una respuesta estable y coherente con la clase real del billete. Sin embargo, en la condición de desenfoque, los tres modelos clasificaron erróneamente el billete como falso. La CNN obtuvo una confianza de 98%, la CNN-LSTM una confianza de 89% y la CNN-GRU una confianza de 99%. Este comportamiento muestra que el desenfoque redujo la información visual disponible para el reconocimiento, afectando principalmente los contornos, texturas y patrones finos del billete. Asimismo, en la condición de baja resolución, los tres modelos también clasificaron el billete verdadero como falso. La CNN presentó una confianza de 100%, la CNN-LSTM de 99% y la CNN-GRU de 100%. Este resultado evidencia que la pérdida de resolución tiene un efecto negativo en la clasificación, debido a que elimina detalles relevantes que los modelos utilizan para diferenciar billetes auténticos y falsificados.

La Figura 46 resume la robustez de los modelos ante las condiciones de captura evaluadas. La CNN presentó una consistencia de 57% respecto a la predicción original y una confianza media de 92%. La CNN-LSTM alcanzó una consistencia de 86% y una confianza media de 91%. Por su parte, la CNN-GRU obtuvo una consistencia de 100% y una confianza media de 95%, lo que indica una respuesta más estable en términos de comparación con su predicción original.

Figura 46

Robustez ante condiciones de captura de un Billete verdadero de 100 soles




```
Resumen — IMG_2922.jpg (mejor modelo: CNN_GRU)
  CNN        | Original: VERDADERO (96.1%) | consistencia 57% | confianza media 92%
  CNN_LSTM   | Original: FALSO (98.4%) | consistencia 86% | confianza media 91%
  CNN_GRU    | Original: FALSO (99.9%) | consistencia 100% | confianza media 95%
```

Nota. Elaboración propia

A partir de los resultados, se observa que la CNN-GRU presentó la mayor consistencia y confianza media. No obstante, debe señalarse que su predicción original para esta prueba aparece como falso, pese a tratarse de un billete verdadero. Por tanto, aunque el modelo mantuvo estabilidad en sus respuestas, dicha estabilidad no necesariamente implica una clasificación correcta en todos los casos. Este aspecto resulta relevante, debido a que una alta consistencia puede representar robustez, pero también puede mantener un error si la predicción inicial fue incorrecta. En contraste, la CNN y la CNN-LSTM reconocieron el billete original como verdadero, aunque fueron afectadas por condiciones críticas como desenfoque y baja resolución. Esto indica que ambos modelos responden adecuadamente cuando la imagen conserva suficiente información visual, pero disminuyen su desempeño cuando se degradan los rasgos espaciales del billete.

5.1.2. Prueba para un billete de 100 soles – enfoque falso

En esta prueba se evaluó el comportamiento de los modelos CNN, CNN-LSTM y CNN-GRU frente a una imagen correspondiente a un billete falso de 100 soles. El propósito fue observar si las arquitecturas entrenadas podían identificar correctamente la falsificación cuando la imagen era sometida a distintas condiciones de captura. Para ello, se consideraron las mismas alteraciones aplicadas en la prueba anterior: imagen original, baja luz, sobreexposición, desenfoque, ruido, rotación y baja resolución.

La Figura 47 muestra el billete falso de 100 soles bajo las condiciones mencionadas. En la imagen original se aprecia una captura con iluminación moderada y una estructura general reconocible del billete. Sin embargo, al aplicar baja luz, los detalles visuales se reducen y algunas zonas pierden contraste. En la condición de sobreexposición, la imagen presenta pérdida de información en áreas claras, lo que afecta la lectura de elementos impresos. El desenfoque disminuye la nitidez de los bordes y detalles internos, mientras que el ruido introduce interferencias visuales que alteran la textura del billete. Por otro lado, la rotación cambia la orientación espacial de la imagen, y la baja resolución reduce de manera considerable la definición de los elementos gráficos. Estas variaciones permitieron evaluar la sensibilidad de cada modelo ante imágenes degradadas. En el caso de billetes falsos, este análisis resulta importante porque una clasificación errónea puede permitir que un billete adulterado sea reconocido como verdadero. Por ello, se observó con mayor atención la respuesta de los modelos frente a la clase falso.

Figura 47

Reconocimiento de captura de un Billete falso de 100 soles



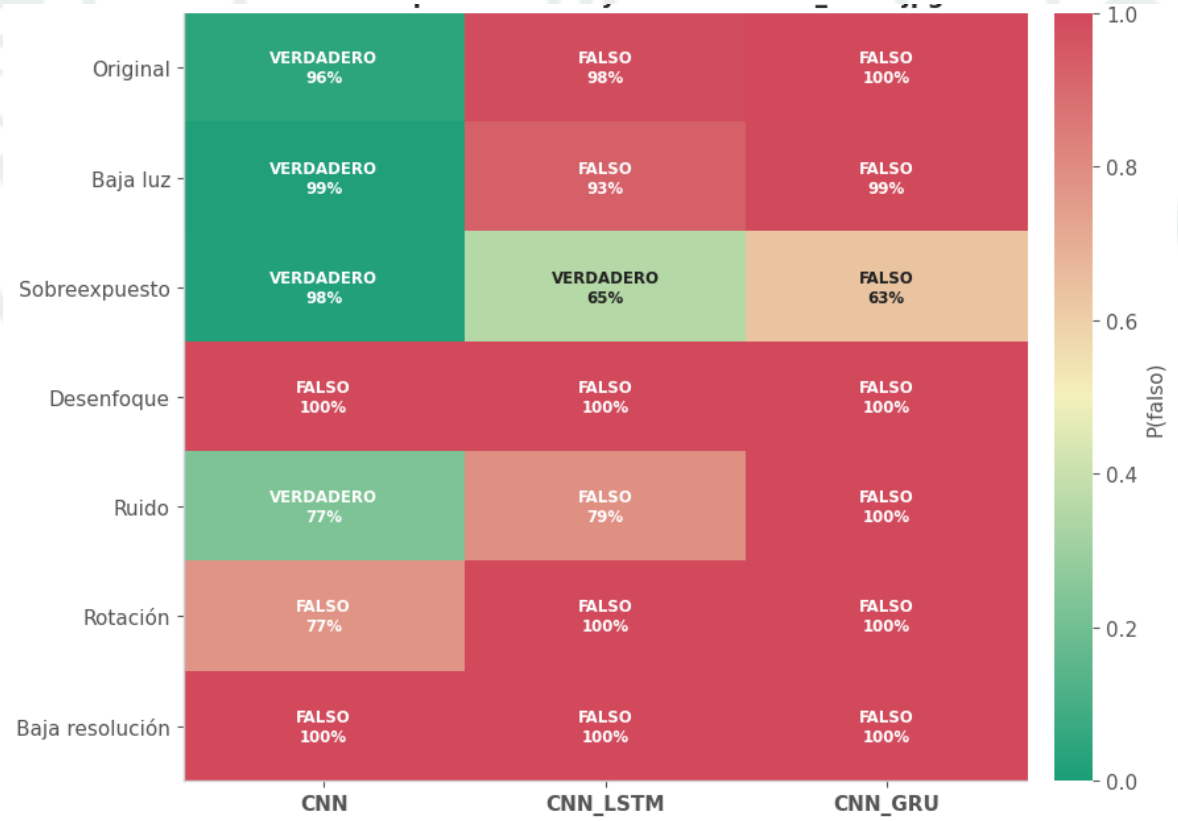


Nota. Elaboración propia

La Figura 48 presenta las predicciones obtenidas por cada arquitectura en las diferentes condiciones de captura.

Figura 48

Predicción por condición y modelo de un Billete falso de 100 soles



Nota. Elaboración propia

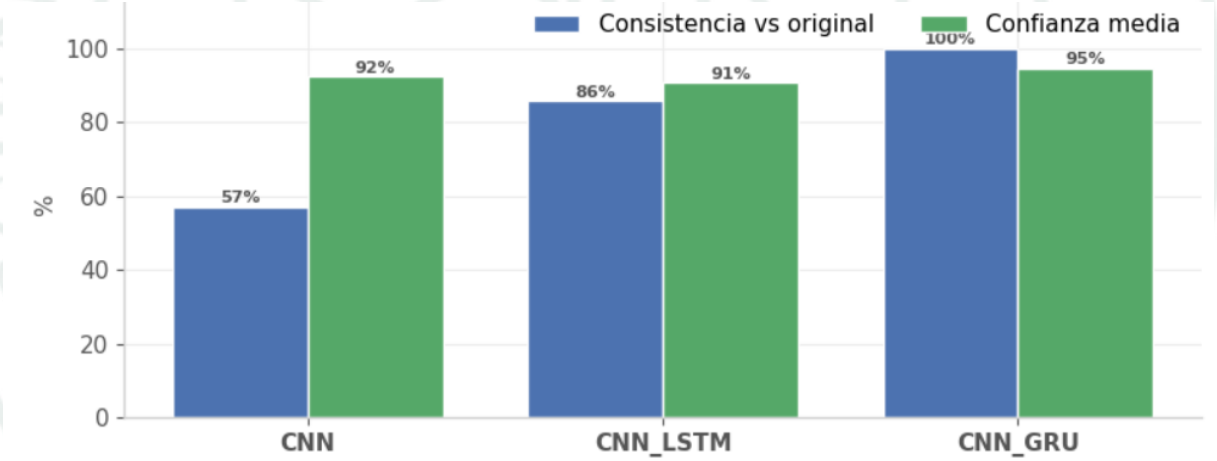
- En la condición original, el modelo CNN clasificó el billete como verdadero con 96%, lo cual representa un error, ya que la imagen correspondía a un billete falso. En cambio, CNN-LSTM lo clasificó como falso con 98% y CNN-GRU también lo clasificó como falso con 100 %. Este resultado muestra que las arquitecturas recurrentes respondieron mejor desde la captura inicial.
- En la condición de baja luz, el modelo CNN volvió a clasificar la imagen como verdadero con 99%, manteniendo el error observado en la imagen original. Por su parte, CNN-LSTM clasificó el billete como falso con 93% y CNN-GRU lo hizo con 99%. Esto evidencia que la reducción de iluminación no afectó de forma crítica a los modelos híbridos, mientras que la CNN presentó dificultad para reconocer la falsificación bajo esta condición.
- En la imagen sobreexpuesta, los resultados fueron más variables. La CNN clasificó el billete como verdadero con 98%, y la CNN-LSTM también lo clasificó como verdadero con 65%. En cambio, CNN-GRU mantuvo la clasificación correcta como falso, aunque con una confianza menor de 63%. Esto indica que la sobreexposición afectó a los tres modelos, pero el CNN-GRU conservó la orientación correcta de la predicción.
- En la condición de desenfoque, los tres modelos clasificaron el billete como falso con 100%. Este resultado muestra que, aunque el desenfoque reduce detalles visuales, en este caso los modelos interpretaron correctamente la imagen como falsificada. A diferencia de la prueba con billete verdadero, el desenfoque no provocó una clasificación incorrecta, sino que reforzó la asociación con la clase falsa.
- En la condición con ruido, la CNN clasificó el billete como verdadero con 77%, lo cual nuevamente representa un error. La CNN-LSTM clasificó la imagen como falso con 79%, mientras que la CNN-GRU mantuvo la clasificación correcta como falso con 100 %. Esto permite observar que el ruido afectó con mayor fuerza al modelo CNN, mientras que las arquitecturas híbridas mostraron mejor respuesta, especialmente CNN-GRU.
- En la condición de rotación, los tres modelos clasificaron el billete como falso. La CNN lo hizo con 77%, mientras que CNN-LSTM y CNN-GRU alcanzaron 100%. Este resultado indica que la variación en la orientación del billete no perjudicó la clasificación de los modelos, y que las redes híbridas mantuvieron mayor seguridad en la predicción.
- En la condición de baja resolución, los tres modelos clasificaron correctamente el billete como falso con 100%. Este comportamiento indica que, aun cuando la imagen perdió definición, los modelos lograron conservar la decisión asociada a falsificación. Sin

embargo, debe considerarse que en otros casos la baja resolución puede afectar la identificación de detalles finos, por lo que su impacto depende de la imagen evaluada y de los rasgos que permanezcan visibles.

La Figura 49 resume la robustez de los modelos ante las condiciones de captura. El modelo CNN presentó una consistencia de 57% y una confianza media de 92%. Este valor indica que la CNN no mantuvo una respuesta estable, ya que clasificó como verdadero varias condiciones que correspondían a un billete falso. La CNN-LSTM alcanzó una consistencia de 86 % y una confianza media de 91%, mostrando una mejora frente a la CNN, aunque todavía presentó error en la condición de sobreexposición. Finalmente, la CNN-GRU obtuvo una consistencia de 100% y una confianza media de 95%, siendo el modelo con mejor comportamiento en esta prueba.

Figura 49

Robustez ante condiciones de captura de un Billete falso de 100 soles




```
Resumen — IMG_2922.jpg (mejor modelo: CNN_GRU)
  CNN        | Original: VERDADERO (96.1%) | consistencia 57% | confianza media 92%
  CNN_LSTM   | Original: FALSO (98.4%) | consistencia 86% | confianza media 91%
  CNN_GRU    | Original: FALSO (99.9%) | consistencia 100% | confianza media 95%
```

Nota. Elaboración propia

De acuerdo con el resumen mostrado en la Figura 49, el modelo CNN clasificó la imagen original como verdadero con 96.1%, lo que evidencia una falsa aceptación del billete adulterado. En contraste, CNN-LSTM identificó el billete original como falso con 98.4%, y CNN-GRU lo clasificó como falso con 99.9%. Este resultado confirma que las arquitecturas

híbridas, principalmente CNN-GRU, tuvieron mejor capacidad para reconocer la falsificación en el billete evaluado.

En síntesis, la prueba realizada con el billete falso de 100 soles demuestra que el modelo CNN presentó mayor dificultad para reconocer la falsificación en condiciones como imagen original, baja luz, sobreexposición y ruido. La CNN-LSTM mostró un desempeño más estable, aunque fue afectada por la sobreexposición. Por su parte, CNN-GRU mantuvo la clasificación correcta en todas las condiciones evaluadas, con mayor consistencia y confianza media. Por ello, en esta prueba, CNN-GRU evidenció el mejor comportamiento para la identificación de billetes falsificados bajo variaciones de captura.

CONCLUSIONES

Se logró identificar billetes peruanos mediante arquitecturas de aprendizaje profundo basadas en CNN y CNN-RNN aplicadas a visión por computadora. Los resultados obtenidos permitieron comprobar que las características visuales de los billetes pueden ser procesadas computacionalmente para diferenciar billetes verdaderos y falsificados. En la comparación general, el modelo CNN-GRU presentó el mejor desempeño, con una exactitud de 98.93%, precisión para la clase falso de 99.66%, recall de 98.33%, F1-score de 98.99% y AUC de 0.9993. Por ello, se concluye que la arquitectura híbrida CNN-GRU obtuvo el comportamiento más favorable dentro del conjunto de prueba evaluado.

Se analizaron las características visuales de los billetes peruanos mediante técnicas de procesamiento digital de imágenes, considerando escala de grises, detección de bordes, gradientes, filtros y análisis de textura. Este procedimiento permitió reconocer diferencias en contraste, nitidez, contornos, ruido, iluminación y definición de detalles entre billetes verdaderos y falsos. En ese sentido, las imágenes procesadas ofrecieron entradas adecuadas para la CNN, debido a que conservaron información espacial útil para el reconocimiento de patrones visuales asociados a autenticidad o falsificación.

Se entrenó una red CNN para extraer patrones espaciales relacionados con billetes falsos y auténticos. Durante el entrenamiento, la CNN evidenció una mejora progresiva en la exactitud y una reducción de la pérdida, lo que demostró su capacidad para aprender rasgos visuales como bordes, texturas, zonas de contraste y detalles impresos. En la etapa de prueba, la CNN alcanzó una exactitud de 98.22%, una precisión de 100.00% para la clase falso, un recall de 96.67% y un F1-score de 98.31%. Estos resultados muestran que la CNN logró un rendimiento alto, aunque presentó 10 billetes falsos clasificados erróneamente como verdaderos.

Se implementaron arquitecturas recurrentes CNN-LSTM y CNN-GRU para complementar la clasificación mediante el procesamiento secuencial de las características extraídas por la CNN. La CNN-LSTM alcanzó una exactitud de 98.04%, precisión de 100.00% para la clase falso, recall de 96.33% y F1-score de 98.13%. Por su parte, la CNN-GRU obtuvo mejores resultados, con una exactitud de 98.93%, recall de 98.33% y F1-score de 98.99% para la clase falso. Esto permite concluir que la incorporación de una unidad recurrente tipo GRU aportó un mejor equilibrio entre precisión y sensibilidad frente a la clase falsificada.

Se evaluó el desempeño del modelo híbrido CNN-RNN bajo diferentes condiciones de captura y se comparó con modelos individuales. Los resultados demostraron que las condiciones de iluminación, sobreexposición, desenfoque, ruido, rotación y baja resolución influyen en la clasificación, especialmente cuando se pierde información visual fina del billete. En las pruebas realizadas, el modelo CNN-GRU mostró mayor estabilidad frente a variaciones de captura y menor cantidad de errores en el conjunto de prueba, clasificando correctamente 555 imágenes de 561. Por tanto, se concluye que el modelo híbrido CNN-GRU presentó mejor desempeño general para la identificación de billetes peruanos falsificados en comparación con CNN y CNN-LSTM.

RECOMENDACIONES

1. Ampliar el conjunto de imágenes de billetes peruanos, incorporando más denominaciones, diferentes niveles de desgaste, fondos variados y condiciones reales de captura, permitiría mejorar la capacidad de generalización de los modelos CNN y CNN-RNN.
2. Resulta conveniente mejorar la calidad y control de las imágenes de entrada, principalmente en casos de desenfoque y baja resolución, debido a que estas condiciones afectaron la clasificación y pueden generar errores en la identificación de billetes verdaderos o falsificados.
3. Para futuras pruebas, conviene utilizar la arquitectura CNN-GRU como base, debido a que presentó el mejor desempeño general en el conjunto de prueba, con mayor exactitud, recall y F1-score para la clase falso.
4. Sería útil aplicar técnicas adicionales de preprocesamiento digital de imágenes, como realce de contraste, reducción de ruido y normalización de iluminación, para fortalecer la extracción de características visuales antes del ingreso de las imágenes al modelo.
5. Evaluar el modelo con imágenes capturadas desde dispositivos móviles o cámaras comunes permitiría comprobar su comportamiento en escenarios más cercanos al uso cotidiano en comercios, transporte o puntos de atención.
6. Finalmente, desarrollar una futura implementación funcional en una aplicación o sistema de prueba, considerando una interfaz simple que permita cargar o capturar imágenes de billetes y obtener una clasificación automática como verdadero o falso, ampliaría el alcance práctico de la investigación.

REFERENCIAS BIBLIOGRÁFICAS

Aguilar, E. P., y González, G. F. L. (2021). Delitos monetarios de falsificación y tráfico de billetes en Perú, vinculados a organizaciones criminales. http://repositorio.cidecuador.org/jspui/handle/123456789/1227

Aguilar, J. A. C. (2016). Monedas, billetes y sociedad en el Perú, 1826-1901. In Historia de la moneda en el Perú (pp. 327-364).

Alegre, E., Barreiro, J., Castejón, M., y Suarez, S. (2008). Computer Vision and Classification Techniques on the Surface Finish Control in Machining Processes. Universidad de León.León, España: E.I. Industrial e Informática.

Antonius, F., Ramu, J., Sasikala, P., Sekhar, J. C., y Mary, S. S. C. (2023). DeepCyberDetect: Hybrid AI for Counterfeit Currency Detection with GAN-CNN-RNN using African Buffalo Optimization. International Journal of Advanced Computer Science and Applications, 14(7). https://doi.org/10.14569/ijacsa.2023.0140772

Arana, C. (2021). Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales (No. 797). Serie Documentos de Trabajo.

Barrionuevo Salinas, A. M. (2024). La inaplicación del Artículo 256° del código penal y los daños generados en billetes y monedas en circulación, Huánuco-2021.

Cabezón, M. (2018). Predicción demanda eléctrica española. Implementación de redes neuronales recurrentes en Python.

Casanova, R., y Bazán, D. (2024). Modelo de visión artificial basado en Deep Learning, para detectar actividades delictivas con pistola dentro de Lima Metropolitana. Upc.edu.pe. http://hdl.handle.net/10757/683153

Cerezo Aguirre, S. M., y Fajardo Estrada, C. (2018). La economía de la falsificación de billetes: Teoría económica y evidencia para Bolivia. Revista de Análisis del Banco Central de Bolivia, 29, 127.

Cifuentes, A., Mendoza, E., Lizcano, M., Santrich, A., y Moreno-Trillos, S. (2019). Desarrollo de una red neuronal convolucional para reconocer patrones en imágenes. Investigación y desarrollo en TIC, 10(2), 7-17.

Diaz Holgado, R., y Vilcas Villalba, D. R. (2022). Aplicativo móvil para la detección de la enfermedad del Mosaico común en la yuca (Manihot esculenta) utilizando redes neuronales convolucionales, 2021.

European Central Bank. (2024). Number of counterfeit euro banknotes remains low in 2023. https://www.nbb.be/en/news-events/number-counterfeit-euro-banknotes-withdrawn-circulation-belgium-2024

G4s (2022). Cómo detectar billetes falsos. G4s.com. https://www.g4s.com/es-pe/g4s-te-informa/boletines-informativos-g4s/2022/09/19/como-detectar-billetes-falsos

García, E. M. (2012). Visión Artificial. España: Fundación para la Universitat Orbeta de Cataluya.

Hamed, N., Y Azzo, F. A. (2024). Advanced Methods for Identifying Counterfeit Currency: Using Deep Learning and Machine Learning. NTU Journal of Engineering and Technology, 3(3). https://doi.org/10.56286/ntujet.v3i3.944

Huaytalla Usurin, B. J., y Humari Quispe, D. O. (2020). Desarrollo de un sistema electrónico portátil orientado a la detección de billetes falsos de 50, 100 y 200 soles basado en algoritmos de procesamiento digital de imágenes.http://hdl.handle.net/10757/652400

Loaiza Quintana, A. F., Manzano Herrera, D. A., y Múnera Salazar, L. E. (2012). Sistema de visión artificial para conteo de objetos en movimiento. El Hombre y la Máquina(40), 87-101. Recuperado el 20 de Enero de 2020, de https://www.redalyc.org/articulo.oa?id=47826850010

Maguiña Asis, M. P., y Pascacio Zarzosa, L. F. (2021). Desarrollo de una herramienta digital web para la inspección de puentes, mediante la implementación de redes neuronales CNN y visión por computador para la detección de fisuras y vegetación en dos puentes tipo armadura en Ancash, Perú.http://hdl.handle.net/10757/668053

Ordoñez Zavaleta, M. C. S. (2017). Relación de los sucesos histórico culturales en el cambio conceptual de los spots televisivos del Banco de Crédito del Perú–BCP desarrollados entre 1996 y 2016.

Oscco-Ccuiro, S., Huashuayo-Miranda, E., Y Aquino-Cruz, M. (2024). Modelo de Redes Neuronales Convolucionales para detectar enfermedades en las hojas del cultivo de

Quinua (Chenopodium quinoa) en el Centro Agronómico K'ayra, San Jeronimo, Cusco 2023. C&T Riqchary Revista de investigación en ciencia y tecnología, 6(1), 13-19.

Pachón, C. G., Ballesteros, D. M., y Renza, D. (2023). An efficient deep learning model using network pruning for fake banknote recognition. Expert Systems with Applications, 233, 120961. https://doi.org/10.1016/j.eswa.2023.120961

Porras, J. (2014). CLASIFICATION SYSTEM BASED ON COMPUTER VISION. Escuela Profesional de Ingeniería Electrónica Universidad Ricardo Palma.

Quispe, H, y Ching, S. (2024). Identificación de fibra textil natural mediante visión computacional y técnicas avanzadas de aprendizaje profundo. Upc.edu.pe. http://hdl.handle.net/10757/673411

Robles Pizarro, L. D. (2016). Caracterización y reconocimiento de objetos mediante algoritmos de visión computacional para la interacción de un robot con su entorno.http://hdl.handle.net/20.500.12404/7376

Sadyk, U., Baimukashev, R., y Turan, C. (2024). State-of-the-Art Review of Deep Learning Methods in Fake Banknote Recognition Problem. International Journal of Advanced Computer Science and Applications, 15(1). https://doi.org/10.14569/ijacsa.2024.0150185

Salinas, Y. (2024). Propuesta de modelo predictivo para la detección de fraude en mensajes de texto mediante el uso de Redes Neuronales Recurrentes. Esan.edu.pe; Universidad ESAN. https://hdl.handle.net/20.500.12640/4287

Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.

Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.

Vega de la Cruz, L. E. (2023). Diseño de una máquina clasificadora de truchas en piscigranjas aplicando visión por computadora.http://hdl.handle.net/20.500.12404/24465

Vélez Serrano, J., Moreno Díaz, A., Sanchez Calle, A., & Sanchez-Marín, J. (2003). Visión por Computador. Universidad Rey Juan Carlos, Departamento de Informática, Estadística y Telemática, Móstoles, España.

Vélez Serrano, J., Moreno Díaz, A., Sanchez Calle, A., y Sanchez-Marín, J. (2003). Visión por Computador. Universidad Rey Juan Carlos, Departamento de Informática, Estadística y Telemática, Móstoles, España

Zarate Calderon, G. H. (2022). Análisis de sentimiento en información de medios periodísticos y redes sociales mediante redes neuronales recurrentes.http://hdl.handle.net/20.500.12404/21525

ANEXOS

Anexo N° 1. Cuadro de Operacionalización de variables

Título: "Identificación de billetes falsificados peruanos utilizando arquitecturas híbridas de aprendizaje profundo con CNN y RNN aplicadas a visión por computadora"				
Objetivos	**Hipótesis**	**Variables**	**Indicadores**	**Metodologías**
General	**General**			
Identificar billetes peruanos falsificados mediante arquitecturas de aprendizaje profundo basadas en CNN y CNN-RNN aplicadas a visión por computadora, a fin de evaluar su desempeño en la detección de falsificaciones según sus características visuales.	La aplicación de arquitecturas de aprendizaje profundo basadas en CNN y CNN-RNN, en el marco de la visión por computadora, influye significativamente en la identificación de billetes peruanos falsificados a partir de sus características visuales.	**Variable Independiente** Arquitectura híbrida de aprendizaje profundo CNN–RNN **Variable Dependiente** Identificación de billetes peruanos falsificados	▪ Número de capas utilizadas en la CNN y en la RNN ▪ Exactitud del modelo en el entrenamiento (training accuracy) ▪ Pérdida del modelo durante el entrenamiento (training loss) ▪ Cantidad de características extraídas por la CNN ▪ Precisión en la detección de billetes falsos ▪ Exactitud global del modelo ▪ Sensibilidad para billetes falsos ▪ Tasa de falsos positivos ▪ Tasa de falsos negativos ▪ Matriz de confusión obtenida en la clasificación final	La investigación emplea un enfoque cuantitativo. El estudio es de tipo aplicado. Se utiliza un diseño experimental.
Específicos	**Específicos**			
Analizar las características visuales de los billetes peruanos para definir las entradas que serán procesadas por la CNN.	Las características visuales extraídas de los billetes permiten a la CNN identificar patrones espaciales asociados a billetes falsos y auténticos.			
Entrenar una red CNN para extraer patrones espaciales relacionados con billetes falsos y auténticos.	El entrenamiento de la CNN contribuye a mejorar la detección de diferencias visuales entre billetes legítimos y adulterados.			
Implementar una red RNN que procese información secuencial del billete para complementar la clasificación.	El procesamiento secuencial realizado por la RNN incrementa la precisión al analizar secuencias del billete.			
Evaluar el desempeño del modelo híbrido CNN-RNN en diferentes condiciones de captura y compararlo con modelos individuales.	El modelo híbrido CNN-RNN obtiene un desempeño superior bajo diferentes condiciones de captura en comparación con las redes utilizadas de manera independiente.			

Nota. Elaboración propia

Anexo N° 2. Código del sistema de identificación de billetes falsificados

```
!pip install gradio==3.41.0 -q
!pip install scikit-learn matplotlib seaborn -q

print("="*70)
print("  SISTEMA DE IDENTIFICACIÓN DE BILLETES PERUANOS - v2.0")
print("="*70)
print("\n  Configurando entorno...")

import os, sys, random, warnings, time
from datetime import datetime

warnings.filterwarnings('ignore')
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import matplotlib.cm as cm
from matplotlib.patches import Rectangle, FancyArrowPatch
from matplotlib.colors import LinearSegmentedColormap
import seaborn as sns
plt.style.use('seaborn-v0_8-darkgrid')

import tensorflow as tf
tf.get_logger().setLevel('ERROR')

from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import (Conv2D, MaxPooling2D, Dense, Dropout,
Flatten,
                                     BatchNormalization,     LSTM,     GRU,
Bidirectional,
                                     Reshape, GlobalAveragePooling2D, Input,
                                     TimeDistributed,    Concatenate,    Add,
Lambda,
                                     SeparableConv2D, DepthwiseConv2D)
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import (EarlyStopping, ReduceLROnPlateau,
                                        ModelCheckpoint, CSVLogger)
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
import tensorflow.keras.backend as K

from sklearn.metrics import (confusion_matrix, classification_report,
                             accuracy_score, precision_score, recall_score,
                             f1_score, roc_auc_score, roc_curve,
                             precision_recall_curve,
average_precision_score,
                             cohen_kappa_score, matthews_corrcoef,
                             balanced_accuracy_score, top_k_accuracy_score)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
import pandas as pd
```

```
from itertools import cycle

gpus = tf.config.list_physical_devices('GPU')
if gpus:
    print(f"  GPU detectada: {gpus[0].name}")
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)
else:
    print("  CPU mode - el entrenamiento será más lento")

np.random.seed(42)
tf.random.set_seed(42)
random.seed(42)

print("\n  Todas las librerías cargadas correctamente")
print("="*70)

print("\n" + "="*70)
print("  DESCARGANDO DATASET DE BILLETES PERUANOS")
print("="*70)

if not os.path.exists('/content/Dataset-Billetes-Peruanos'):
    print("  Descargando dataset desde GitHub...")
    os.system('git clone -q https://github.com/NicolasPCS/Dataset-Billetes-
Peruanos.git')
    print("  Dataset descargado")
else:
    print("  Dataset ya existe")

DATASET_PATH = '/content/Dataset-Billetes-Peruanos'
TRAIN_PATH   = os.path.join(DATASET_PATH, 'training_set')
VAL_PATH     = os.path.join(DATASET_PATH, 'validation_set')
TEST_PATH    = os.path.join(DATASET_PATH, 'testing_set')

CATEGORIAS = sorted(os.listdir(TRAIN_PATH))
NUM_CLASES  = len(CATEGORIAS)

def contar_imagenes(path):
    total = 0
    per_cat = {}
    for cat in CATEGORIAS:
        cp = os.path.join(path, cat)
        n      =      len([f      for      f      in      os.listdir(cp)      if
f.lower().endswith(('.jpg','.jpeg','.png'))]) if os.path.exists(cp) else 0
        per_cat[cat] = n
        total += n
    return total, per_cat

n_train, train_per = contar_imagenes(TRAIN_PATH)
n_val,   val_per   = contar_imagenes(VAL_PATH)
n_test,  test_per  = contar_imagenes(TEST_PATH)

print(f"\n  ESTADÍSTICAS DEL DATASET:")
print(f"    Categorías:  {NUM_CLASES}")
print(f"    Training:    {n_train:,} imágenes")
print(f"    Validation:  {n_val:,} imágenes")
print(f"    Testing:     {n_test:,} imágenes")
```

```
print(f"    TOTAL:        {n_train+n_val+n_test:,} imágenes")

fig, axes = plt.subplots(1, 3, figsize=(18, 5))
fig.suptitle("DISTRIBUCIÓN    DE    IMÁGENES    POR    CLASE",    fontsize=15,
fontweight='bold')

for  ax,  (split_name,  per_cat)  in  zip(axes,  [("Training",  train_per),
("Validation", val_per), ("Testing", test_per)]):
    cats = list(per_cat.keys())
    vals = list(per_cat.values())
    colors = plt.cm.tab20(np.linspace(0, 1, len(cats)))
    bars = ax.barh([c[:20] for c in cats], vals, color=colors)
    ax.set_title(f"{split_name} set", fontsize=12, fontweight='bold')
    ax.set_xlabel("Número de imágenes")
    for bar, v in zip(bars, vals):
        ax.text(bar.get_width() + 0.5, bar.get_y() + bar.get_height()/2,
                str(v), va='center', fontsize=8)
    ax.grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.savefig('/content/00_distribucion_clases.png',                    dpi=150,
bbox_inches='tight')
plt.show()
print("  Figura guardada: 00_distribucion_clases.png")
print("="*70)

INFO_BILLETES = {
    '10':  {'nuevo':  {'personaje':  'Chabuca  Granda',  'color_dominante':
'Verde', 'color_rgb': (74,159,74)},
            'antiguo':  {'personaje':  'J.A.  Quiñones',  'color_dominante':
'Azul verdoso', 'color_rgb': (70,130,150)}},
    '20': {'nuevo': {'personaje': 'José M. Arguedas', 'color_dominante':
'Amarillo', 'color_rgb': (204,153,51)},
            'antiguo':    {'personaje':    'R.    Porras    Barrenechea',
'color_dominante': 'Marrón', 'color_rgb': (180,120,60)}},
    '50': {'nuevo': {'personaje': 'María Rostworowski', 'color_dominante':
'Rojo', 'color_rgb': (178,34,34)},
            'antiguo':      {'personaje':      'Abraham      Valdelomar',
'color_dominante': 'Gris violáceo', 'color_rgb': (120,100,140)}},
    '100':{'nuevo': {'personaje': 'Pedro Paulet', 'color_dominante': 'Azul
acero', 'color_rgb': (70,130,180)},
            'antiguo':  {'personaje':  'Jorge  Basadre',  'color_dominante':
'Rojo ladrillo', 'color_rgb': (170,74,68)}}
}

print("  Información de billetes peruanos cargada")
print("="*70)

print("\n" + "="*70)
print("  MÓDULO PDI v2.0 - Incluye análisis LBP, FFT, GLCM, Gradientes ricos")
print("="*70)

class PDI:

    @staticmethod
    def asegurar_uint8(img):
```

```
        return np.clip(img, 0, 255).astype(np.uint8) if img.dtype != np.uint8
else img

    @staticmethod
    def asegurar_rgb(img):
        return cv2.cvtColor(img, cv2.COLOR_GRAY2RGB) if len(img.shape) == 2
else img

    @staticmethod
    def normalizar(img):
        if img.max() > img.min():
            return ((img - img.min()) / (img.max() - img.min()) *
255).astype(np.uint8)
        return np.zeros_like(img, dtype=np.uint8)

    @staticmethod
    def a_gris(img):      return cv2.cvtColor(img, cv2.COLOR_RGB2GRAY) if
len(img.shape)==3 else img
    @staticmethod
    def a_hsv(img):    return cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    @staticmethod
    def a_lab(img):    return cv2.cvtColor(img, cv2.COLOR_RGB2LAB)
    @staticmethod
    def a_ycrcb(img):  return cv2.cvtColor(img, cv2.COLOR_RGB2YCrCb)
    @staticmethod
    def a_xyz(img):    return cv2.cvtColor(img, cv2.COLOR_RGB2XYZ)

    @staticmethod
    def canal_rojo(img):
        c=np.zeros_like(img); c[:,:,0]=img[:,:,0]; return c
    @staticmethod
    def canal_verde(img):
        c=np.zeros_like(img); c[:,:,1]=img[:,:,1]; return c
    @staticmethod
    def canal_azul(img):
        c=np.zeros_like(img); c[:,:,2]=img[:,:,2]; return c
    @staticmethod
    def canal_h(img): return cv2.cvtColor(img,cv2.COLOR_RGB2HSV)[:,:,0]
    @staticmethod
    def canal_s(img): return cv2.cvtColor(img,cv2.COLOR_RGB2HSV)[:,:,1]
    @staticmethod
    def canal_v(img): return cv2.cvtColor(img,cv2.COLOR_RGB2HSV)[:,:,2]
    @staticmethod
    def canal_l(img): return cv2.cvtColor(img,cv2.COLOR_RGB2LAB)[:,:,0]
    @staticmethod
    def canal_a_lab(img): return cv2.cvtColor(img,cv2.COLOR_RGB2LAB)[:,:,1]
    @staticmethod
    def canal_b_lab(img): return cv2.cvtColor(img,cv2.COLOR_RGB2LAB)[:,:,2]

    @staticmethod
    def filtro_media(img, k=5):
        k = k if k%2==1 else k+1; return cv2.blur(img,(k,k))
    @staticmethod
    def filtro_gaussiano(img, k=5, sigma=0):
        k = k if k%2==1 else k+1; return cv2.GaussianBlur(img,(k,k),sigma)
    @staticmethod
    def filtro_mediana(img, k=5):
```

```
        k = k if k%2==1 else k+1; return cv2.medianBlur(img,k)
    @staticmethod
    def filtro_bilateral(img, d=9, sc=75, ss=75):
        return cv2.bilateralFilter(img,d,sc,ss)
    @staticmethod
    def filtro_nlm(img, h=10):
        if len(img.shape)==3:
            return cv2.fastNlMeansDenoisingColored(img,None,h,h,7,21)
        return cv2.fastNlMeansDenoising(img,None,h,7,21)

    @staticmethod
    def ecualizar_histograma(img):
        if len(img.shape)==3:
            yuv=cv2.cvtColor(img,cv2.COLOR_RGB2YUV);
yuv[:,:,0]=cv2.equalizeHist(yuv[:,:,0])
            return cv2.cvtColor(yuv,cv2.COLOR_YUV2RGB)
        return cv2.equalizeHist(img)
    @staticmethod
    def clahe(img, clip_limit=2.0, tile=8):
        c=cv2.createCLAHE(clipLimit=clip_limit,tileGridSize=(tile,tile))
        if len(img.shape)==3:
            lab=cv2.cvtColor(img,cv2.COLOR_RGB2LAB);
lab[:,:,0]=c.apply(lab[:,:,0])
            return cv2.cvtColor(lab,cv2.COLOR_LAB2RGB)
        return c.apply(img)
    @staticmethod
    def correccion_gamma(img, gamma=1.0):
        t=np.array([((i/255)**(      1/gamma))*255      for      i      in
range(256)],dtype=np.uint8)
        return cv2.LUT(img,t)
    @staticmethod
    def stretching_contraste(img, p_low=2, p_high=98):
        if len(img.shape)==3:
            r=np.zeros_like(img)
            for i in range(3):
                p2,p98=np.percentile(img[:,:,i],(p_low,p_high))
                if    p98>p2:    r[:,:,i]=np.clip((img[:,:,i]-p2)*255/(p98-
p2),0,255)
            return r.astype(np.uint8)
        p2,p98=np.percentile(img,(p_low,p_high))
        return   np.clip((img-p2)*255/(p98-p2),0,255).astype(np.uint8)   if
p98>p2 else img

    @staticmethod
    def unsharp_masking(img, sigma=1.0, strength=1.5):
        b=cv2.GaussianBlur(img,(0,0),sigma)
        return                  np.clip(cv2.addWeighted(img,1+strength,b,-
strength,0),0,255).astype(np.uint8)
    @staticmethod
    def high_boost(img, k=1.5):
        b=cv2.GaussianBlur(img,(5,5),0)
        return    np.clip(img.astype(np.float32)+k*(img.astype(np.float32)-
b.astype(np.float32)),0,255).astype(np.uint8)

    @staticmethod
    def   _gray(img):   return   cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)   if
len(img.shape)==3 else img
```

```
    @staticmethod
    def canny(img, t1=50, t2=150): return cv2.Canny(PDI._gray(img),t1,t2)
    @staticmethod
    def sobel(img, k=3):
        g=PDI._gray(img)
        sx=cv2.Sobel(g,cv2.CV_64F,1,0,ksize=k);
sy=cv2.Sobel(g,cv2.CV_64F,0,1,ksize=k)
        return PDI.normalizar(np.sqrt(sx**2+sy**2))
    @staticmethod
    def sobel_x(img, k=3):
        return
PDI.normalizar(np.abs(cv2.Sobel(PDI._gray(img),cv2.CV_64F,1,0,ksize=k)))
    @staticmethod
    def sobel_y(img, k=3):
        return
PDI.normalizar(np.abs(cv2.Sobel(PDI._gray(img),cv2.CV_64F,0,1,ksize=k)))
    @staticmethod
    def laplaciano(img, k=3):
        return
PDI.normalizar(np.abs(cv2.Laplacian(PDI._gray(img),cv2.CV_64F,ksize=k)))
    @staticmethod
    def prewitt(img):
        g=PDI._gray(img)
        kx=np.array([[1,0,-1],[1,0,-1],[1,0,-1]],np.float32)
        ky=np.array([[1,1,1],[0,0,0],[-1,-1,-1]],np.float32)
        return                          PDI.normalizar(np.sqrt(cv2.filter2D(g,-
1,kx).astype(np.float32)**2+
                                    cv2.filter2D(g,-
1,ky).astype(np.float32)**2))
    @staticmethod
    def scharr(img):
        g=PDI._gray(img)
        return
PDI.normalizar(np.sqrt(cv2.Scharr(g,cv2.CV_64F,1,0)**2+cv2.Scharr(g,cv2.CV_
64F,0,1)**2))
    @staticmethod
    def roberts(img):
        g=PDI._gray(img).astype(np.float32)
        kx=np.array([[1,0],[0,-1]],np.float32);        ky=np.array([[0,1],[-
1,0]],np.float32)
        return                          PDI.normalizar(np.sqrt(cv2.filter2D(g,-
1,kx)**2+cv2.filter2D(g,-1,ky)**2))
    @staticmethod
    def log(img, sigma=1.0):
        b=cv2.GaussianBlur(PDI._gray(img),(0,0),sigma)
        return PDI.normalizar(np.abs(cv2.Laplacian(b,cv2.CV_64F)))

    @staticmethod
    def umbral_simple(img, t=127):
        _,b=cv2.threshold(PDI._gray(img),t,255,cv2.THRESH_BINARY); return b
    @staticmethod
    def otsu(img):

_,b=cv2.threshold(PDI._gray(img),0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU);
return b
    @staticmethod
```

```
    def adaptativo_gaussian(img, bs=11, C=2):
        bs=bs if bs%2==1 else bs+1
        return
cv2.adaptiveThreshold(PDI._gray(img),255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2
.THRESH_BINARY,bs,C)
    @staticmethod
    def adaptativo_mean(img, bs=11, C=2):
        bs=bs if bs%2==1 else bs+1
        return
cv2.adaptiveThreshold(PDI._gray(img),255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THR
ESH_BINARY,bs,C)

    @staticmethod
    def _morph(img, op, k=3):
        kernel=np.ones((k,k),np.uint8)
        return cv2.morphologyEx(PDI._gray(img),op,kernel)
    @staticmethod
    def erosion(img,k=3,it=1):
        return
cv2.erode(PDI._gray(img),np.ones((k,k),np.uint8),iterations=it)
    @staticmethod
    def dilatacion(img,k=3,it=1):
        return
cv2.dilate(PDI._gray(img),np.ones((k,k),np.uint8),iterations=it)
    @staticmethod
    def apertura(img,k=3):    return PDI._morph(img,cv2.MORPH_OPEN,k)
    @staticmethod
    def cierre(img,k=3):     return PDI._morph(img,cv2.MORPH_CLOSE,k)
    @staticmethod
    def                gradiente_morfologico(img,k=3):                return
PDI._morph(img,cv2.MORPH_GRADIENT,k)
    @staticmethod
    def top_hat(img,k=9):    return PDI._morph(img,cv2.MORPH_TOPHAT,k)
    @staticmethod
    def black_hat(img,k=9):  return PDI._morph(img,cv2.MORPH_BLACKHAT,k)

    @staticmethod
    def detectar_contornos(img):
        g=PDI._gray(img); _,b=cv2.threshold(g,127,255,cv2.THRESH_BINARY)

contours,_=cv2.findContours(b,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
        r=np.zeros((*g.shape,3),dtype=np.uint8)
        cv2.drawContours(r,contours,-1,(0,255,0),2); return r
    @staticmethod
    def esquinas_harris(img, bs=2, ks=3, k=0.04):
        g=PDI._gray(img).astype(np.float32)
        r=img.copy()          if          len(img.shape)==3          else
cv2.cvtColor(img,cv2.COLOR_GRAY2RGB)
        dst=cv2.cornerHarris(g,bs,ks,k); dst=cv2.dilate(dst,None)
        r[dst>0.01*dst.max()]=[255,0,0]; return r
    @staticmethod
    def detector_orb(img, n=500):
        g=PDI._gray(img)
        r=img.copy()          if          len(img.shape)==3          else
cv2.cvtColor(img,cv2.COLOR_GRAY2RGB)
        orb=cv2.ORB_create(nfeatures=n); kps=orb.detect(g,None)
```

```
cv2.drawKeypoints(r,kps,r,color=(0,255,0),flags=cv2.DRAW_MATCHES_FLAGS_DRAW
_RICH_KEYPOINTS)
        return r
    @staticmethod
    def hough_lineas(img, threshold=100):
        g=PDI._gray(img)
        r=img.copy()           if           len(img.shape)==3           else
cv2.cvtColor(img,cv2.COLOR_GRAY2RGB)
        edges=cv2.Canny(g,50,150)

lines=cv2.HoughLinesP(edges,1,np.pi/180,threshold,minLineLength=50,maxLineG
ap=10)
        if lines is not None:
            for l in lines:
                x1,y1,x2,y2=l[0]; cv2.line(r,(x1,y1),(x2,y2),(255,0,0),2)
        return r

    @staticmethod
    def fft_magnitud(img):

        g=PDI._gray(img).astype(np.float32)
        f=np.fft.fft2(g); fshift=np.fft.fftshift(f)
        mag=20*np.log(np.abs(fshift)+1)
        return PDI.normalizar(mag)

    @staticmethod
    def lbp_textura(img, P=8, R=1):

        g=PDI._gray(img)
        h,w=g.shape; lbp=np.zeros_like(g,dtype=np.uint8)
        angles=np.linspace(0,2*np.pi,P,endpoint=False)
        for i in range(R,h-R):
            for j in range(R,w-R):
                center=g[i,j]; code=0
                for k_,(a) in enumerate(angles):
                    xi=int(np.round(i-R*np.sin(a)));
xj=int(np.round(j+R*np.cos(a)))
                    xi=np.clip(xi,0,h-1); xj=np.clip(xj,0,w-1)
                    if g[xi,xj]>=center: code|=(1<<k_)
                lbp[i,j]=code
        return lbp

    @staticmethod
    def lbp_rapido(img):

        g=PDI._gray(img).astype(np.float32)
        kernels=[
            np.array([[0,0,0],[0,-1,1],[0,0,0]],np.float32),
            np.array([[0,0,0],[0,-1,0],[0,0,1]],np.float32),
            np.array([[0,0,0],[0,-1,0],[0,1,0]],np.float32),
            np.array([[0,0,0],[1,-1,0],[0,0,0]],np.float32),
        ]
        result=np.zeros_like(g)
        for kk in kernels:
            result+=np.abs(cv2.filter2D(g,-1,kk))
        return PDI.normalizar(result)
```

```
    @staticmethod
    def gradiente_orientacion(img):

        g=PDI._gray(img).astype(np.float32)
        gx=cv2.Sobel(g,cv2.CV_64F,1,0,ksize=3)
        gy=cv2.Sobel(g,cv2.CV_64F,0,1,ksize=3)
        angle=np.arctan2(gy,gx)*180/np.pi+180
        return PDI.normalizar(angle)

    @staticmethod
    def mapa_entropia_local(img, block=8):

        g=PDI._gray(img)
        h,w=g.shape; result=np.zeros_like(g,dtype=np.float32)
        for i in range(0,h-block,block):
            for j in range(0,w-block,block):
                blk=g[i:i+block,j:j+block].flatten()
                hist,_=np.histogram(blk,bins=16,range=(0,255))
                p=hist/(hist.sum()+1e-10)
                ent=-np.sum(p*np.log2(p+1e-10))
                result[i:i+block,j:j+block]=ent
        return PDI.normalizar(result)

    @staticmethod
    def mapa_calor_densidad(img):

        edges=PDI.canny(img).astype(np.float32)
        heat=cv2.GaussianBlur(edges,(21,21),0)
        heat_norm=PDI.normalizar(heat)
        return cv2.applyColorMap(heat_norm, cv2.COLORMAP_JET)

    @staticmethod
    def analisis_frecuencia_color(img):

        result=np.zeros_like(img)
        for i in range(3):
            canal=img[:,:,i].astype(np.float32)
            f=np.fft.fft2(canal); fshift=np.fft.fftshift(f)
            mag=20*np.log(np.abs(fshift)+1)
            result[:,:,i]=PDI.normalizar(mag)
        return result

    @staticmethod
    def extraer_caracteristicas(img):
        g=PDI._gray(img)
        sx=cv2.Sobel(g,cv2.CV_64F,1,0,ksize=3);
sy=cv2.Sobel(g,cv2.CV_64F,0,1,ksize=3)
        hist=cv2.calcHist([g],[0],None,[256],[0,256]).flatten()
        hn=hist/(hist.sum()+1e-10)
        return {
            'Media': round(np.mean(g),2), 'Desviación': round(np.std(g),2),
            'Varianza':   round(np.var(g),2),   'Mínimo':   int(np.min(g)),
'Máximo': int(np.max(g)),
            'Contraste': int(np.max(g)-np.min(g)),
            'Entropía': round(-np.sum(hn*np.log2(hn+1e-10)),4),
            'Energía': round(np.mean(np.sqrt(sx**2+sy**2)),2),
```

```
            'Kurtosis':     round(float(np.mean(((g-np.mean(g))/np.std(g+1e-
10))**4)),4),
            'Asimetría':    round(float(np.mean(((g-np.mean(g))/np.std(g+1e-
10))**3)),4),
        }

    @staticmethod
    def calcular_histograma_rgb(img):
        return {c: cv2.calcHist([img],[i],None,[256],[0,256]).flatten() for
i,c in enumerate(['R','G','B'])}

    @staticmethod
    def redimensionar(img, size=(224,224)):
        return cv2.resize(img, size, interpolation=cv2.INTER_AREA)

print(f"  Clase PDI v2.0 cargada con técnicas extendidas (LBP, FFT, Entropia
local, etc.)")
print("="*70)

def cargar_imagen(categoria, dataset_path=TRAIN_PATH, indice=None):
    cp=os.path.join(dataset_path, categoria)
    if not os.path.exists(cp): return None
    imgs=[f          for          f          in          os.listdir(cp)          if
f.lower().endswith(('.jpg','.jpeg','.png'))]
    if not imgs: return None
    nombre=random.choice(imgs)      if      indice      is      None      else
imgs[min(indice,len(imgs)-1)]
    img=cv2.imread(os.path.join(cp,nombre))
    if img is None: return None
    return cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

def mostrar_imagen(img, titulo="Imagen", figsize=(10,6)):
    plt.figure(figsize=figsize)
    plt.imshow(img, cmap='gray' if len(img.shape)==2 else None)
    plt.title(titulo,fontsize=14,fontweight='bold'); plt.axis('off')
    plt.tight_layout(); plt.show()

def crear_grid_imagenes(imagenes, titulos, filas, columnas, figsize=(16,12),
titulo_principal=None):
    fig,axes=plt.subplots(filas,columnas,figsize=figsize)
    if                                                  titulo_principal:
fig.suptitle(titulo_principal,fontsize=16,fontweight='bold',y=1.02)
    axes=axes.flatten() if filas*columnas>1 else [axes]
    for i,(img,titulo) in enumerate(zip(imagenes,titulos)):
        if i<len(axes) and img is not None:
            axes[i].imshow(img,cmap='gray' if len(img.shape)==2 else None)
            axes[i].set_title(titulo,fontsize=10,fontweight='bold')
        if i<len(axes): axes[i].axis('off')
    for i in range(len(imagenes),len(axes)): axes[i].axis('off')
    plt.tight_layout(); plt.show(); return fig

print("  Funciones auxiliares cargadas")
print("="*70)

print("\n" + "="*70)
print("  OBJETIVO A: ANÁLISIS ENRIQUECIDO DE CARACTERÍSTICAS VISUALES")
print("  Para definir las entradas que procesará la CNN")
```

```
print("="*70)

categoria_demo = CATEGORIAS[0]
img_demo = cargar_imagen(categoria_demo)
img_demo = PDI.redimensionar(img_demo, (400, 200))

print(f"  Billete seleccionado: {categoria_demo}")
print(f"  Dimensiones: {img_demo.shape}")

stats = PDI.extraer_caracteristicas(img_demo)
print("\n  Características estadísticas:")
for k,v in stats.items(): print(f"    • {k}: {v}")

fig = plt.figure(figsize=(22, 14))
fig.suptitle("ANÁLISIS DE ESPACIOS DE COLOR PARA CLASIFICACIÓN DE BILLETES",
             fontsize=16, fontweight='bold', y=0.98)

slots = [
    ("RGB Original", img_demo, None),
    ("Canal R", img_demo[:,:,0], 'Reds'),
    ("Canal G", img_demo[:,:,1], 'Greens'),
    ("Canal B", img_demo[:,:,2], 'Blues'),
    ("Escala Grises", PDI.a_gris(img_demo), 'gray'),
    ("HSV - Hue (tono)", PDI.canal_h(img_demo), 'hsv'),
    ("HSV - Saturación", PDI.canal_s(img_demo), 'plasma'),
    ("HSV - Value", PDI.canal_v(img_demo), 'gray'),
    ("LAB - Luminosidad L", PDI.canal_l(img_demo), 'gray'),
    ("LAB - Canal A (rojo-verde)", PDI.canal_a_lab(img_demo), 'RdYlGn'),
    ("LAB - Canal B (azul-amarillo)", PDI.canal_b_lab(img_demo), 'coolwarm'),
    ("YCrCb", cv2.cvtColor(img_demo, cv2.COLOR_RGB2YCrCb), None),
]

for idx, (titulo, img, cmap) in enumerate(slots):
    ax = fig.add_subplot(3, 4, idx+1)
    ax.imshow(img, cmap=cmap)
    ax.set_title(titulo, fontsize=10, fontweight='bold', pad=4)
    ax.axis('off')
    if len(img.shape)==2:
        ax.set_xlabel(f"μ={img.mean():.1f}  σ={img.std():.1f}", fontsize=8,
labelpad=2)

plt.tight_layout()
plt.savefig('/content/A1_espacios_color.png', dpi=150, bbox_inches='tight')
plt.show()
print("  Figura A1 guardada: A1_espacios_color.png")

fig = plt.figure(figsize=(22, 18))
fig.suptitle("CARACTERÍSTICAS AVANZADAS PARA DETECCIÓN DE FALSIFICACIONES",
             fontsize=16, fontweight='bold', y=0.99)

adv_slots = [
    ("Original", img_demo, None),
    ("CLAHE (contraste local)", PDI.clahe(img_demo), None),
    ("Unsharp Masking (nitidez)", PDI.unsharp_masking(img_demo), None),
    ("Stretching de Contraste", PDI.stretching_contraste(img_demo), None),
    ("Canny (bordes)", PDI.canny(img_demo,50,150), 'gray'),
    ("Sobel (magnitud gradiente)", PDI.sobel(img_demo), 'hot'),
```

```
    ("Laplaciano (cambios 2do orden)", PDI.laplaciano(img_demo), 'gray'),
    ("Scharr (gradiente preciso)", PDI.scharr(img_demo), 'inferno'),
    ("Gradiente Orientación", PDI.gradiente_orientacion(img_demo), 'hsv'),
    ("FFT Magnitud (frecuencias)", PDI.fft_magnitud(img_demo), 'viridis'),
    ("FFT Color (patrones cromáticos)",
PDI.analisis_frecuencia_color(img_demo), None),
    ("LBP Textura (patrones locales)", PDI.lbp_rapido(img_demo), 'gray'),
    ("Mapa Entropía Local", PDI.mapa_entropia_local(img_demo), 'plasma'),
    ("Mapa Calor Bordes", PDI.mapa_calor_densidad(img_demo), None),
    ("Top Hat (detalles brillantes)", PDI.top_hat(img_demo), 'gray'),
    ("Black Hat (detalles oscuros)", PDI.black_hat(img_demo), 'gray'),
    ("Adaptativo Gaussiano", PDI.adaptativo_gaussian(img_demo), 'gray'),
    ("Binarización Otsu", PDI.otsu(img_demo), 'gray'),
    ("Detector ORB (puntos clave)", PDI.detector_orb(img_demo,300), None),
    ("Gradiente Morfológico", PDI.gradiente_morfologico(img_demo), 'gray'),
]

for idx, (titulo, img, cmap) in enumerate(adv_slots):
    ax = fig.add_subplot(4, 5, idx+1)
    ax.imshow(img, cmap=cmap)
    ax.set_title(titulo, fontsize=9, fontweight='bold', pad=3)
    ax.axis('off')
    cat_colors = {'Original': 'black', 'CLAHE':'orange', 'Unsharp':'orange',
                  'Stretching':'orange', 'Canny':'red', 'Sobel':'red',
                  'Laplaciano':'red',       'Scharr':'red',       'Gradiente
O':'purple',
                  'FFT':'blue', 'LBP':'green', 'Mapa':'brown'}
    for key,col in cat_colors.items():
        if key.lower() in titulo.lower():
            for spine in ax.spines.values():
                spine.set_edgecolor(col); spine.set_linewidth(3)
            break

legend_items = [plt.Line2D([0],[0],color=c,lw=3,label=l) for c,l in [
    ('orange','Mejora contraste/nitidez'), ('red','Detección de bordes'),
    ('purple','Orientación/gradiente'), ('blue','Dominio frecuencia (FFT)'),
    ('green','Textura (LBP)'), ('brown','Análisis estadístico')]]
fig.legend(handles=legend_items, loc='lower center', ncol=3, fontsize=10,
           bbox_to_anchor=(0.5,-0.01), framealpha=0.9)

plt.tight_layout()
plt.savefig('/content/A2_caracteristicas_avanzadas.png',           dpi=150,
bbox_inches='tight')
plt.show()
print("  Figura A2 guardada: A2_caracteristicas_avanzadas.png")

denominaciones = ['10', '20', '50', '100']
tecnicas = [
    ("Original",      lambda i: i,                                  None),
    ("CLAHE",        lambda i: PDI.clahe(i),                        None),
    ("Bordes Canny",  lambda i: PDI.canny(i),                      'gray'),
    ("Gradiente",     lambda i: PDI.sobel(i),                       'hot'),
    ("FFT",                                  lambda   i:   PDI.fft_magnitud(i),
'viridis'),
    ("LBP     Textura",               lambda      i:      PDI.lbp_rapido(i),
'gray'),
```

```
    ("Mapa Entropía", lambda i: PDI.mapa_entropia_local(i),
'plasma'),
    ("Mapa Calor", lambda i: PDI.mapa_calor_densidad(i),
None),
]

fig, axes = plt.subplots(4, 8, figsize=(26, 14))
fig.suptitle("COMPARATIVA DE CARACTERÍSTICAS POR DENOMINACIÓN DE BILLETE\n"
             "Análisis multidimensional para detección de falsificaciones",
             fontsize=15, fontweight='bold')

for j, (nom,_,_) in enumerate(tecnicas):
    axes[0,j].set_title(nom, fontsize=10, fontweight='bold', color='navy',
pad=8)

for i, denom in enumerate(denominaciones):
    cat = next((c for c in CATEGORIAS if c.startswith(denom)), None)
    if cat:
        img = cargar_imagen(cat)
        if img is not None:
            img = PDI.redimensionar(img, (350,175))
            axes[i,0].set_ylabel(f"S/ {denom}", fontsize=13,
fontweight='bold',
                                 rotation=0, ha='right', va='center',
color='darkred')
            for j, (_, fn, cmap) in enumerate(tecnicas):
                res = fn(img)
                axes[i,j].imshow(res, cmap=cmap)
                axes[i,j].axis('off')
                data = res if len(res.shape)==2 else PDI.a_gris(res)
                axes[i,j].set_xlabel(f"μ={data.mean():.0f}", fontsize=7,
labelpad=1)

plt.tight_layout()
plt.savefig('/content/A3_comparativa_denominaciones.png', dpi=150,
bbox_inches='tight')
plt.show()
print("  Figura A3 guardada: A3_comparativa_denominaciones.png")

fig, axes = plt.subplots(4, 4, figsize=(20, 16))
fig.suptitle("ANÁLISIS ESTADÍSTICO DE DISTRIBUCIONES DE COLOR\n"
             "Firma cromática de cada denominación",
             fontsize=14, fontweight='bold')

for i, denom in enumerate(denominaciones):
    cat = next((c for c in CATEGORIAS if c.startswith(denom)), None)
    if cat:
        img = cargar_imagen(cat)
        if img is not None:
            img = PDI.redimensionar(img,(400,200))
            axes[i,0].imshow(img); axes[i,0].axis('off')
            axes[i,0].set_title(f"S/ {denom}", fontsize=12,
fontweight='bold', color='darkred')
            for ci,(cname,col) in
enumerate([('R','red'),('G','green'),('B','blue')]):
                hist=cv2.calcHist([img],[ci],None,[256],[0,256]).flatten()
```

```
axes[i,1].plot(hist,color=col,alpha=0.8,label=cname,linewidth=1.5)
                axes[i,1].fill_between(range(256),hist,alpha=0.1,color=col)
            axes[i,1].set_title("Histograma
RGB",fontsize=10,fontweight='bold')
            axes[i,1].legend(fontsize=8);       axes[i,1].set_xlim([0,255]);
axes[i,1].grid(alpha=0.3)
            axes[i,1].set_xlabel("Intensidad");
axes[i,1].set_ylabel("Frecuencia")
            hsv=PDI.a_hsv(img)
            for                          ci,(cname,col)                          in
enumerate([('H','purple'),('S','orange'),('V','gray')]):
                hist=cv2.calcHist([hsv],[ci],None,[256],[0,256]).flatten()

axes[i,2].plot(hist,color=col,alpha=0.8,label=cname,linewidth=1.5)
            axes[i,2].set_title("Histograma
HSV",fontsize=10,fontweight='bold')
            axes[i,2].legend(fontsize=8);       axes[i,2].set_xlim([0,255]);
axes[i,2].grid(alpha=0.3)
            axes[i,2].set_xlabel("Intensidad");
axes[i,2].set_ylabel("Frecuencia")
            lbp=PDI.lbp_rapido(img)
            hist_lbp,_=np.histogram(lbp.flatten(),bins=64,range=(0,255))

axes[i,3].bar(range(64),hist_lbp,color='teal',alpha=0.7,width=1)
            axes[i,3].set_title("Histograma                                LBP
(Textura)",fontsize=10,fontweight='bold')
            axes[i,3].set_xlabel("Bin                                LBP");
axes[i,3].set_ylabel("Frecuencia")
            axes[i,3].grid(alpha=0.3)

plt.tight_layout()
plt.savefig('/content/A4_histogramas_enriquecidos.png',              dpi=150,
bbox_inches='tight')
plt.show()
print("  Figura A4 guardada: A4_histogramas_enriquecidos.png")

print("\n  ANÁLISIS DE CARACTERÍSTICAS COMPLETADO")
print("  4 figuras generadas con análisis multidimensional")
print("="*70)

print("\n" + "="*70)
print("  DEMOSTRACIÓN COMPLETA DE TÉCNICAS PDI")
print("="*70)

fig, axes = plt.subplots(2,4,figsize=(18,9))
fig.suptitle('FILTROS      DE      SUAVIZADO      -      Reducción      de
ruido',fontsize=14,fontweight='bold')
for i,(titulo,img) in enumerate([
    ("Original",img_demo), ("Media 5x5",PDI.filtro_media(img_demo,5)),
    ("Media          9x9",PDI.filtro_media(img_demo,9)),          ("Gaussiano
5x5",PDI.filtro_gaussiano(img_demo,5)),
    ("Gaussiano       9x9",PDI.filtro_gaussiano(img_demo,9)),       ("Mediana
5x5",PDI.filtro_mediana(img_demo,5)),
    ("Bilateral",PDI.filtro_bilateral(img_demo)),                ("Non-Local
Means",PDI.filtro_nlm(img_demo))]):
```

```
    axes[i//4,i%4].imshow(img);
axes[i//4,i%4].set_title(titulo,fontsize=11,fontweight='bold')
    axes[i//4,i%4].axis('off')
plt.tight_layout();
plt.savefig('/content/PDI1_suavizado.png',dpi=150,bbox_inches='tight');
plt.show()
print("  Figura guardada: PDI1_suavizado.png")

fig, axes = plt.subplots(2,4,figsize=(18,9))
fig.suptitle('MEJORA DE CONTRASTE - Realce de
detalles',fontsize=14,fontweight='bold')
for i,(titulo,img) in enumerate([
    ("Original",img_demo), ("Ecualización
Histograma",PDI.ecualizar_histograma(img_demo)),
    ("CLAHE clip=2",PDI.clahe(img_demo,2.0)), ("CLAHE
clip=4",PDI.clahe(img_demo,4.0)),
    ("Gamma 0.5",PDI.correccion_gamma(img_demo,0.5)), ("Gamma
1.5",PDI.correccion_gamma(img_demo,1.5)),
    ("Gamma 2.0",PDI.correccion_gamma(img_demo,2.0)),
("Stretching",PDI.stretching_contraste(img_demo))]):
    axes[i//4,i%4].imshow(img);
axes[i//4,i%4].set_title(titulo,fontsize=11,fontweight='bold')
    axes[i//4,i%4].axis('off')
plt.tight_layout();
plt.savefig('/content/PDI2_contraste.png',dpi=150,bbox_inches='tight');
plt.show()
print("  Figura guardada: PDI2_contraste.png")

fig, axes = plt.subplots(3,4,figsize=(18,13))
fig.suptitle('DETECCIÓN DE BORDES - Crucial para
falsificaciones',fontsize=14,fontweight='bold')
for i,(titulo,img,cmap) in enumerate([
    ("Original",img_demo,None), ("Grises",PDI.a_gris(img_demo),'gray'),
    ("Canny t50-150",PDI.canny(img_demo,50,150),'gray'), ("Canny t100-
200",PDI.canny(img_demo,100,200),'gray'),
    ("Sobel magnitud",PDI.sobel(img_demo),'hot'), ("Sobel
X",PDI.sobel_x(img_demo),'gray'),
    ("Sobel Y",PDI.sobel_y(img_demo),'gray'),
("Laplaciano",PDI.laplaciano(img_demo),'gray'),
    ("Prewitt",PDI.prewitt(img_demo),'gray'),
("Scharr",PDI.scharr(img_demo),'inferno'),
    ("Roberts",PDI.roberts(img_demo),'gray'),
("LoG",PDI.log(img_demo),'gray')]):
    axes[i//4,i%4].imshow(img,cmap=cmap);
axes[i//4,i%4].set_title(titulo,fontsize=10,fontweight='bold')
    axes[i//4,i%4].axis('off')
plt.tight_layout();
plt.savefig('/content/PDI3_bordes.png',dpi=150,bbox_inches='tight');
plt.show()
print("  Figura guardada: PDI3_bordes.png")

fig, axes = plt.subplots(2,4,figsize=(18,9))
fig.suptitle('OPERACIONES MORFOLÓGICAS - Análisis de
formas',fontsize=14,fontweight='bold')
for i,(titulo,img) in enumerate([
    ("Original (grises)",PDI.asegurar_rgb(PDI.a_gris(img_demo))),
    ("Erosión",PDI.asegurar_rgb(PDI.erosion(img_demo))),
```

```
    ("Dilatación",PDI.asegurar_rgb(PDI.dilatacion(img_demo))),
    ("Apertura",PDI.asegurar_rgb(PDI.apertura(img_demo))),
    ("Cierre",PDI.asegurar_rgb(PDI.cierre(img_demo))),
    ("Gradiente
Morfológico",PDI.asegurar_rgb(PDI.gradiente_morfologico(img_demo))),
    ("Top Hat",PDI.asegurar_rgb(PDI.top_hat(img_demo))),
    ("Black Hat",PDI.asegurar_rgb(PDI.black_hat(img_demo)))]):
    axes[i//4,i%4].imshow(img);
axes[i//4,i%4].set_title(titulo,fontsize=10,fontweight='bold')
    axes[i//4,i%4].axis('off')
plt.tight_layout();
plt.savefig('/content/PDI4_morfologia.png',dpi=150,bbox_inches='tight');
plt.show()
print("  Figura guardada: PDI4_morfologia.png")

print("  Técnicas PDI visualizadas correctamente")
print("="*70)

print("\n" + "="*70)
print("  OBJETIVOS B y C: ARQUITECTURAS CNN y CNN-RNN")
print("="*70)

IMG_SIZE   = 128
BATCH_SIZE = 32

def crear_modelo_cnn(num_clases, img_size=IMG_SIZE, nombre='CNN_Billetes'):

    inp = Input(shape=(img_size, img_size, 3), name='entrada_imagen')

    x                                                              =
Conv2D(32,(3,3),activation='relu',padding='same',name='conv1_1')(inp)
    x           =           BatchNormalization()(x);           x           =
Conv2D(32,(3,3),activation='relu',padding='same',name='conv1_2')(x)
    x   =   BatchNormalization()(x);   x   =   MaxPooling2D((2,2))(x);   x   =
Dropout(0.25)(x)

    x = Conv2D(64,(3,3),activation='relu',padding='same',name='conv2_1')(x)
    x           =           BatchNormalization()(x);           x           =
Conv2D(64,(3,3),activation='relu',padding='same',name='conv2_2')(x)
    x   =   BatchNormalization()(x);   x   =   MaxPooling2D((2,2))(x);   x   =
Dropout(0.25)(x)

    x = Conv2D(128,(3,3),activation='relu',padding='same',name='conv3_1')(x)
    x           =           BatchNormalization()(x);           x           =
Conv2D(128,(3,3),activation='relu',padding='same',name='conv3_2')(x)
    x   =   BatchNormalization()(x);   x   =   MaxPooling2D((2,2))(x);   x   =
Dropout(0.25)(x)

    x = Conv2D(256,(3,3),activation='relu',padding='same',name='conv4_1')(x)
    x = BatchNormalization()(x)
    x = GlobalAveragePooling2D()(x)

    x = Dense(256,activation='relu')(x); x = Dropout(0.5)(x)
    x = Dense(128,activation='relu')(x); x = Dropout(0.3)(x)
    salida = Dense(num_clases,activation='softmax',name='clasificacion')(x)

    model = Model(inp, salida, name=nombre)
```

```
    model.compile(optimizer=Adam(1e-4), loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model

def crear_modelo_cnn_lstm(num_clases, img_size=IMG_SIZE,
nombre='CNN_LSTM_Hibrido'):

    inp = Input(shape=(img_size, img_size, 3), name='entrada_imagen')

    x = Conv2D(32,(3,3),activation='relu',padding='same')(inp); x =
BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)
    x = Conv2D(64,(3,3),activation='relu',padding='same')(x); x =
BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)
    x = Conv2D(128,(3,3),activation='relu',padding='same')(x); x =
BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)

    shp = K.int_shape(x)
    seq_len = shp[1] * shp[2]
    x = Reshape((seq_len, shp[3]))(x)

    x = Bidirectional(LSTM(64, return_sequences=True))(x); x =
Dropout(0.3)(x)
    x = Bidirectional(LSTM(32, return_sequences=False))(x); x =
Dropout(0.3)(x)

    x = Dense(128,activation='relu')(x); x = Dropout(0.5)(x)
    salida = Dense(num_clases,activation='softmax',name='clasificacion')(x)

    model = Model(inp, salida, name=nombre)
    model.compile(optimizer=Adam(1e-4), loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model

def crear_modelo_cnn_gru(num_clases, img_size=IMG_SIZE,
nombre='CNN_GRU_Hibrido'):

    inp = Input(shape=(img_size, img_size, 3), name='entrada_imagen')

    x = Conv2D(32,(3,3),activation='relu',padding='same')(inp); x =
BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)
    x = Conv2D(64,(3,3),activation='relu',padding='same')(x); x =
BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)
    x = Conv2D(128,(3,3),activation='relu',padding='same')(x); x =
BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)

    shp = K.int_shape(x); seq_len = shp[1]*shp[2]
    x = Reshape((seq_len, shp[3]))(x)

    x = Bidirectional(GRU(64, return_sequences=True))(x); x = Dropout(0.3)(x)
    x = Bidirectional(GRU(32, return_sequences=False))(x); x =
Dropout(0.3)(x)
```

```
    x = Dense(128,activation='relu')(x); x = Dropout(0.5)(x)
    salida = Dense(num_clases,activation='softmax',name='clasificacion')(x)

    model = Model(inp, salida, name=nombre)
    model.compile(optimizer=Adam(1e-4), loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model

print("\n  ARQUITECTURA 1: CNN Puro")
m1=crear_modelo_cnn(NUM_CLASES); m1.summary()

print("\n  ARQUITECTURA 2: CNN-LSTM Bidireccional")
m2=crear_modelo_cnn_lstm(NUM_CLASES); m2.summary()

print("\n  ARQUITECTURA 3: CNN-GRU Bidireccional")
m3=crear_modelo_cnn_gru(NUM_CLASES); m3.summary()

print("\n  COMPARATIVA DE PARÁMETROS:")
print(f"  {'Modelo':<20} {'Parámetros':>15} {'Parámetros MB':>15}")
print("  " + "-"*52)
for m,n in [(m1,'CNN'),(m2,'CNN-LSTM'),(m3,'CNN-GRU')]:
    p=m.count_params(); mb=p*4/1024/1024
    print(f"  {n:<20} {p:>15,} {mb:>14.2f} MB")

del m1,m2,m3
import gc; gc.collect()
print("\n  Arquitecturas definidas correctamente")
print("="*70)

print("\n" + "="*70)
print("  PREPARACIÓN DE DATOS PARA ENTRENAMIENTO")
print("="*70)

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=12,
    width_shift_range=0.12,
    height_shift_range=0.12,
    shear_range=0.08,
    zoom_range=0.12,
    brightness_range=[0.75, 1.25],
    channel_shift_range=15,
    horizontal_flip=False,
    fill_mode='nearest'
)

val_datagen  = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    TRAIN_PATH, target_size=(IMG_SIZE,IMG_SIZE), batch_size=BATCH_SIZE,
    class_mode='categorical', shuffle=True)

val_generator = val_datagen.flow_from_directory(
    VAL_PATH, target_size=(IMG_SIZE,IMG_SIZE), batch_size=BATCH_SIZE,
    class_mode='categorical', shuffle=False)
```

```
test_generator = test_datagen.flow_from_directory(
    TEST_PATH, target_size=(IMG_SIZE,IMG_SIZE), batch_size=BATCH_SIZE,
    class_mode='categorical', shuffle=False)

class_indices = train_generator.class_indices
idx_to_class  = {v:k for k,v in class_indices.items()}

print(f"  Training:    {train_generator.samples:,} muestras")
print(f"  Validation:  {val_generator.samples:,} muestras")
print(f"  Testing:     {test_generator.samples:,} muestras")
print(f"  Clases:      {len(class_indices)}")
print(f"  Batch size:  {BATCH_SIZE}")
print(f"  IMG_SIZE:    {IMG_SIZE}x{IMG_SIZE}")

fig, axes = plt.subplots(3, 6, figsize=(18,9))
fig.suptitle("EJEMPLOS DE DATA AUGMENTATION
APLICADO",fontsize=14,fontweight='bold')
batch_x, batch_y = next(train_generator)
for i in range(18):
    ax=axes[i//6, i%6]
    ax.imshow(batch_x[i%len(batch_x)])
    cls=idx_to_class[np.argmax(batch_y[i%len(batch_y)])]
    ax.set_title(cls[:15],fontsize=8); ax.axis('off')
plt.tight_layout()
plt.savefig('/content/DATA_augmentation.png',dpi=150,bbox_inches='tight');
plt.show()
print("  Figura guardada: DATA_augmentation.png")
print("="*70)

print("\n" + "="*70)
print("  ENTRENAMIENTO DE MODELOS")
print("="*70)

EPOCHS     = 30
PATIENCE_ES = 8
PATIENCE_LR = 4

def get_callbacks(nombre):
    return [
        EarlyStopping(monitor='val_loss', patience=PATIENCE_ES,
                      restore_best_weights=True, verbose=1),
        ReduceLROnPlateau(monitor='val_loss', factor=0.5,
patience=PATIENCE_LR,
                          verbose=1, min_lr=1e-7),
        CSVLogger(f'/content/log_{nombre}.csv'),
    ]

resultados = {}

for nombre, fn_crear in [("CNN", crear_modelo_cnn),
                         ("CNN-LSTM", crear_modelo_cnn_lstm),
                         ("CNN-GRU",  crear_modelo_cnn_gru)]:
    print(f"\n  Entrenando {nombre}...")
    print("  " + "-"*40)
    t0 = time.time()
    modelo = fn_crear(NUM_CLASES)
```

```
    hist   = modelo.fit(train_generator, validation_data=val_generator,
                        epochs=EPOCHS,      callbacks=get_callbacks(nombre),
verbose=1)
    elapsed = time.time() - t0
    resultados[nombre] = {
        'modelo': modelo, 'historia': hist.history,
        'train_acc': hist.history['accuracy'][-1],
        'val_acc':   hist.history['val_accuracy'][-1],
        'tiempo_min': elapsed/60
    }
    print(f"  {nombre} → Train: {resultados[nombre]['train_acc']:.4f} | Val:
{resultados[nombre]['val_acc']:.4f} | Tiempo: {elapsed/60:.1f} min")

print("\n  ENTRENAMIENTO COMPLETADO")
print("="*70)

print("\n" + "="*70)
print("  OBJETIVO D: EVALUACIÓN EXHAUSTIVA DEL DESEMPEÑO")
print("  Accuracy, Precision, Recall, F1, AUC-ROC, Kappa, MCC, Top-2")
print("="*70)

evaluaciones = {}

for nombre, data in resultados.items():
    modelo = data['modelo']
    test_generator.reset()

    loss, accuracy = modelo.evaluate(test_generator, verbose=0)

    test_generator.reset()
    y_pred_proba = modelo.predict(test_generator, verbose=0)
    y_pred = np.argmax(y_pred_proba, axis=1)
    y_true = test_generator.classes[:len(y_pred)]

    y_true_bin = label_binarize(y_true, classes=list(range(NUM_CLASES)))

    top2 = top_k_accuracy_score(y_true, y_pred_proba[:len(y_true)], k=2)

    try:
        auc_roc = roc_auc_score(y_true_bin, y_pred_proba[:len(y_true)],
                                multi_class='ovr', average='macro')
    except Exception:
        auc_roc = float('nan')

    evaluaciones[nombre] = {
        'accuracy':         accuracy,
        'balanced_acc':     balanced_accuracy_score(y_true, y_pred),
        'top2_acc':         top2,
        'loss':             loss,
        'precision_macro':  precision_score(y_true, y_pred, average='macro',
zero_division=0),
        'recall_macro':       recall_score(y_true,  y_pred,  average='macro',
zero_division=0),
        'f1_macro':             f1_score(y_true,  y_pred,  average='macro',
zero_division=0),
        'f1_weighted':         f1_score(y_true,  y_pred,  average='weighted',
zero_division=0),
```

```
            'auc_roc':          auc_roc,
            'cohen_kappa':      cohen_kappa_score(y_true, y_pred),
            'mcc':              matthews_corrcoef(y_true, y_pred),
            'y_true':           y_true,
            'y_pred':           y_pred,
            'y_pred_proba':     y_pred_proba,
            'y_true_bin':       y_true_bin,
            'tiempo_min':       data['tiempo_min']
        }

        print(f"\n  {nombre}:")
        print(f"    Accuracy:           {accuracy:.4f}  ({accuracy*100:.2f}%)")
        print(f"                                        Balanced          Accuracy:
{evaluaciones[nombre]['balanced_acc']:.4f}")
        print(f"    Top-2 Accuracy:     {top2:.4f}")
        print(f"    Loss:               {loss:.4f}")
        print(f"                                        Precision           (macro):
{evaluaciones[nombre]['precision_macro']:.4f}")
        print(f"                                           Recall           (macro):
{evaluaciones[nombre]['recall_macro']:.4f}")
        print(f"    F1 (macro):         {evaluaciones[nombre]['f1_macro']:.4f}")
        print(f"                                            F1           (weighted):
{evaluaciones[nombre]['f1_weighted']:.4f}")
        print(f"    AUC-ROC (macro):    {evaluaciones[nombre]['auc_roc']:.4f}")
        print(f"                                          Cohen           Kappa:
{evaluaciones[nombre]['cohen_kappa']:.4f}")
        print(f"    MCC:                {evaluaciones[nombre]['mcc']:.4f}")
        print(f"    Tiempo entren.:     {evaluaciones[nombre]['tiempo_min']:.1f}
min")

print("\n" + "="*70)
print("  TABLA COMPARATIVA FINAL DE MODELOS")
print("="*70)

cols      =      ["Accuracy","Bal.Acc","Top-2","F1-macro","F1-weighted","AUC-
ROC","Kappa","MCC"]
header = f"{'Modelo':<12}" + "".join(f"{c:>12}" for c in cols)
print(header)
print("-"*len(header))
for nombre, m in evaluaciones.items():
    vals = [m['accuracy'], m['balanced_acc'], m['top2_acc'],
            m['f1_macro'], m['f1_weighted'], m['auc_roc'],
            m['cohen_kappa'], m['mcc']]
    row = f"{nombre:<12}" + "".join(f"{v:>12.4f}" for v in vals)
    print(row)

mejor_modelo        =        max(evaluaciones,        key=lambda        x:
evaluaciones[x]['f1_weighted'])
print(f"\n  MEJOR MODELO (por F1-weighted): {mejor_modelo}")
print(f"    F1-weighted: {evaluaciones[mejor_modelo]['f1_weighted']:.4f}")
print(f"    Accuracy:    {evaluaciones[mejor_modelo]['accuracy']*100:.2f}%")
print(f"    AUC-ROC:     {evaluaciones[mejor_modelo]['auc_roc']:.4f}")

categorias_radar    =    ['Accuracy','Bal.Acc','Top-2    Acc','F1-macro','F1-
weighted','AUC-ROC','Kappa','MCC (norm)']
N = len(categorias_radar)
angulos = np.linspace(0, 2*np.pi, N, endpoint=False).tolist()
```

```
angulos += angulos[:1]

fig, ax = plt.subplots(1,1,figsize=(10,10),subplot_kw=dict(polar=True))
colores_radar = {'CNN':'blue', 'CNN-LSTM':'green', 'CNN-GRU':'red'}

for nombre, m in evaluaciones.items():
    mcc_norm = (m['mcc']+1)/2
    vals = [m['accuracy'], m['balanced_acc'], m['top2_acc'],
            m['f1_macro'], m['f1_weighted'],
            m['auc_roc'] if not np.isnan(m['auc_roc']) else 0,
            (m['cohen_kappa']+1)/2, mcc_norm]
    vals += vals[:1]
    ax.plot(angulos,   vals,   color=colores_radar[nombre],   linewidth=2.5,
label=nombre)
    ax.fill(angulos, vals, color=colores_radar[nombre], alpha=0.1)

ax.set_xticks(angulos[:-1]);            ax.set_xticklabels(categorias_radar,
fontsize=11)
ax.set_ylim(0,1); ax.set_title("Radar de Métricas - Comparativa de Modelos",
                               fontsize=14, fontweight='bold', pad=25)
ax.legend(loc='upper right', bbox_to_anchor=(1.3,1.1), fontsize=12)
ax.grid(True, alpha=0.4)
plt.tight_layout()
plt.savefig('/content/EVAL1_radar_metricas.png',                    dpi=150,
bbox_inches='tight'); plt.show()
print("\n  Figura guardada: EVAL1_radar_metricas.png")

print("="*70)

print("\n" + "="*70)
print("  CURVAS DE ENTRENAMIENTO")
print("="*70)

colores = {'CNN':'royalblue', 'CNN-LSTM':'seagreen', 'CNN-GRU':'crimson'}

fig, axes = plt.subplots(2, 3, figsize=(20,11))
fig.suptitle("CURVAS     DE     ENTRENAMIENTO     -     COMPARATIVA     DE
MODELOS",fontsize=16,fontweight='bold')

for i, (nombre, data) in enumerate(resultados.items()):
    hist = data['historia']; col = colores[nombre]
    epocas = range(1, len(hist['accuracy'])+1)

    ax = axes[0,i]
    ax.plot(epocas, hist['accuracy'],     color=col, lw=2.5, label='Train')
    ax.plot(epocas,   hist['val_accuracy'],   color=col,   lw=2.5,   ls='--',
label='Validation')
    for                                e,(tr,va)                              in
enumerate(zip(hist['accuracy'],hist['val_accuracy']),1):
        if tr-va > 0.1:
            ax.axvspan(e-0.5, e+0.5, alpha=0.05, color='red')
    ax.fill_between(epocas, hist['accuracy'], hist['val_accuracy'],
                    alpha=0.07, color=col)
    ax.set_title(f"{nombre}                                                  –
Accuracy",fontsize=12,fontweight='bold',color=col)
    ax.set_xlabel("Época"); ax.set_ylabel("Accuracy")
    ax.legend(); ax.grid(alpha=0.3); ax.set_ylim([0,1])
```

```
    ax.axhline(y=evaluaciones[nombre]['accuracy'],color=col,ls=':'
               ,lw=1.5,label=f"Test:
{evaluaciones[nombre]['accuracy']:.3f}")
    ax.legend(fontsize=9)

    ax = axes[1,i]
    ax.plot(epocas, hist['loss'],     color=col, lw=2.5, label='Train')
    ax.plot(epocas,    hist['val_loss'],    color=col,    lw=2.5,    ls='--',
label='Validation')
    ax.fill_between(epocas,  hist['loss'],  hist['val_loss'],  alpha=0.07,
color=col)
    ax.set_title(f"{nombre} — Loss",fontsize=12,fontweight='bold',color=col)
    ax.set_xlabel("Época"); ax.set_ylabel("Loss")
    ax.legend(); ax.grid(alpha=0.3)

plt.tight_layout()
plt.savefig('/content/EVAL2_curvas_entrenamiento.png',              dpi=150,
bbox_inches='tight')
plt.show()
print("  Figura guardada: EVAL2_curvas_entrenamiento.png")

fig, axes = plt.subplots(1,2,figsize=(16,6))
fig.suptitle("CONVERGENCIA        COMPARADA        DE        LOS        3
MODELOS",fontsize=14,fontweight='bold')
for nombre, data in resultados.items():
    h=data['historia'];                                col=colores[nombre];
ep=range(1,len(h['accuracy'])+1)
    axes[0].plot(ep, h['val_accuracy'], color=col, lw=2.5, label=nombre)
    axes[1].plot(ep, h['val_loss'],     color=col, lw=2.5, label=nombre)
for ax, ylabel, title in zip(axes, ['Val Accuracy','Val Loss'],
                             ['Accuracy    de    Validación','Loss    de
Validación']):
    ax.set_xlabel("Época"); ax.set_ylabel(ylabel)
    ax.set_title(title, fontsize=12, fontweight='bold')
    ax.legend(fontsize=11); ax.grid(alpha=0.3)
plt.tight_layout()
plt.savefig('/content/EVAL3_convergencia_comparada.png',dpi=150,bbox_inches
='tight'); plt.show()
print("  Figura guardada: EVAL3_convergencia_comparada.png")
print("="*70)

print("\n" + "="*70)
print("  MATRICES DE CONFUSIÓN DETALLADAS")
print("="*70)

fig, axes = plt.subplots(1, 3, figsize=(22, 7))
fig.suptitle("MATRICES                    DE                    CONFUSIÓN
NORMALIZADAS",fontsize=15,fontweight='bold')

for ax, (nombre, m) in zip(axes, evaluaciones.items()):
    cm = confusion_matrix(m['y_true'], m['y_pred'])
    cm_norm = cm.astype('float') / (cm.sum(axis=1)[:,np.newaxis]+1e-10)
    im = ax.imshow(cm_norm, cmap='Blues', vmin=0, vmax=1, aspect='auto')
    ax.set_title(f"{nombre}\nAcc={m['accuracy']*100:.1f}%
F1={m['f1_weighted']:.3f}  κ={m['cohen_kappa']:.3f}",
                 fontsize=11, fontweight='bold')
```

```
    ax.set_xlabel("Clase     Predicha",fontsize=10);     ax.set_ylabel("Clase
Real",fontsize=10)
    ticks=list(range(len(class_indices)))
    lbls=[idx_to_class[i][:12] for i in ticks]
    ax.set_xticks(ticks); ax.set_xticklabels(lbls,rotation=90,fontsize=7)
    ax.set_yticks(ticks); ax.set_yticklabels(lbls,fontsize=7)
    plt.colorbar(im, ax=ax, fraction=0.046, pad=0.04)
    for i in range(cm_norm.shape[0]):
        for j in range(cm_norm.shape[1]):
            val=cm_norm[i,j]
            ax.text(j,i,f"{val:.2f}",ha='center',va='center',
                    fontsize=5,color='white' if val>0.5 else 'black')

plt.tight_layout()
plt.savefig('/content/EVAL4_matrices_confusion.png',dpi=150,bbox_inches='ti
ght'); plt.show()
print("  Figura guardada: EVAL4_matrices_confusion.png")

print(f"\n  REPORTE DETALLADO - {mejor_modelo}:")
m = evaluaciones[mejor_modelo]
print(classification_report(m['y_true'], m['y_pred'],
                            target_names=[idx_to_class[i]    for    i    in
range(NUM_CLASES)],
                            zero_division=0))
print("="*70)

print("\n" + "="*70)
print("  CURVAS ROC Y PRECISIÓN-RECALL")
print("="*70)

fig, axes = plt.subplots(2, 3, figsize=(20, 12))
fig.suptitle("CURVAS        ROC        Y        PRECISIÓN-RECALL        POR
MODELO",fontsize=15,fontweight='bold')

for col_i, (nombre, m) in enumerate(evaluaciones.items()):
    y_true_bin = m['y_true_bin']
    y_score    = m['y_pred_proba'][:len(m['y_true'])]

    ax_roc = axes[0, col_i]
    for i in range(NUM_CLASES):
        fpr, tpr, _ = roc_curve(y_true_bin[:,i], y_score[:,i])
        try:
            auc_i = roc_auc_score(y_true_bin[:,i], y_score[:,i])
        except Exception:
            auc_i = float('nan')
        ax_roc.plot(fpr, tpr, lw=1, alpha=0.6,
                    label=f"{idx_to_class[i][:10]} ({auc_i:.2f})")
    ax_roc.plot([0,1],[0,1],'k--',lw=1.5,label='Aleatorio')
    auc_macro = m['auc_roc']
    ax_roc.set_title(f"{nombre} — Curva ROC\nAUC-macro={auc_macro:.3f}",
                     fontsize=11,fontweight='bold')
    ax_roc.set_xlabel("Tasa  Falsos  Positivos");  ax_roc.set_ylabel("Tasa
Verdaderos Positivos")
    ax_roc.legend(fontsize=6, loc='lower right'); ax_roc.grid(alpha=0.3)
    ax_roc.set_xlim([0,1]); ax_roc.set_ylim([0,1.02])

    ax_pr = axes[1, col_i]
```

```
    for i in range(NUM_CLASES):
        prec, rec, _ = precision_recall_curve(y_true_bin[:,i], y_score[:,i])
        ap = average_precision_score(y_true_bin[:,i], y_score[:,i])
        ax_pr.plot(rec, prec, lw=1, alpha=0.6,
                   label=f"{idx_to_class[i][:10]} (AP={ap:.2f})")
    mean_ap      =      np.nanmean([average_precision_score(y_true_bin[:,i],
y_score[:,i])
                           for i in range(NUM_CLASES)])
    ax_pr.set_title(f"{nombre} — Precisión-Recall\nmAP={mean_ap:.3f}",
                    fontsize=11,fontweight='bold')
    ax_pr.set_xlabel("Recall"); ax_pr.set_ylabel("Precisión")
    ax_pr.legend(fontsize=6, loc='upper right'); ax_pr.grid(alpha=0.3)
    ax_pr.set_xlim([0,1]); ax_pr.set_ylim([0,1.02])

plt.tight_layout()
plt.savefig('/content/EVAL5_roc_pr.png',dpi=150,bbox_inches='tight');
plt.show()
print("  Figura guardada: EVAL5_roc_pr.png")
print("="*70)

print("\n" + "="*70)
print("  ANÁLISIS DE ERRORES Y EJEMPLOS DE PREDICCIONES")
print("="*70)

fig, axes = plt.subplots(1, 3, figsize=(18, 5))
fig.suptitle("DISTRIBUCIÓN            DE            CONFIANZA            DE
PREDICCIONES",fontsize=14,fontweight='bold')

for ax, (nombre, m) in zip(axes, evaluaciones.items()):
    confianza = m['y_pred_proba'].max(axis=1)[:len(m['y_true'])]
    correcto  = (m['y_pred'] == m['y_true']).astype(bool)
    ax.hist(confianza[correcto],     bins=20,   color='green',   alpha=0.6,
label=f'Correcto (n={correcto.sum()})')
    ax.hist(confianza[~correcto],  bins=20,  color='red',       alpha=0.6,
label=f'Incorrecto (n={(~correcto).sum()})')
    ax.set_title(f"{nombre}",fontsize=12,fontweight='bold')
    ax.set_xlabel("Confianza de predicción"); ax.set_ylabel("Frecuencia")
    ax.legend(); ax.grid(alpha=0.3)
    ax.axvline(confianza[correcto].mean(),  color='green', ls=':', lw=2,
               label=f"μ correcto={confianza[correcto].mean():.2f}")
    ax.axvline(confianza[~correcto].mean(), color='red',   ls=':', lw=2,
               label=f"μ error={confianza[~correcto].mean():.2f}")
    ax.legend(fontsize=8)

plt.tight_layout()
plt.savefig('/content/EVAL6_distribucion_confianza.png',dpi=150,bbox_inches
='tight'); plt.show()
print("  Figura guardada: EVAL6_distribucion_confianza.png")

m = evaluaciones[mejor_modelo]
f1_por_clase    =    f1_score(m['y_true'],    m['y_pred'],    average=None,
zero_division=0)
prec_clase      =  precision_score(m['y_true'],  m['y_pred'],  average=None,
zero_division=0)
rec_clase         =  recall_score(m['y_true'],  m['y_pred'],  average=None,
zero_division=0)
```

```
fig, axes = plt.subplots(1,3,figsize=(20,6))
fig.suptitle(f"MÉTRICAS                POR                CLASE                —
{mejor_modelo}",fontsize=14,fontweight='bold')
nombres_clase = [idx_to_class[i][:18] for i in range(NUM_CLASES)]
x = np.arange(len(nombres_clase))

for ax, vals, title, color in zip(axes,
    [prec_clase, rec_clase, f1_por_clase],
    ["Precisión por clase","Recall por clase","F1-Score por clase"],
    ['steelblue','seagreen','crimson']):
    bars = ax.bar(x, vals, color=color, alpha=0.8, edgecolor='white')
    ax.set_xticks(x);     ax.set_xticklabels(nombres_clase,     rotation=90,
fontsize=8)
    ax.set_title(title,fontsize=12,fontweight='bold'); ax.set_ylim([0,1])
    ax.axhline(y=vals.mean(), color='black', ls='--', lw=1.5,
               label=f"Media={vals.mean():.3f}")
    ax.legend(); ax.grid(axis='y', alpha=0.3)
    for bar,v in zip(bars,vals):
        if v > 0.05:
            ax.text(bar.get_x()+bar.get_width()/2, bar.get_height()+0.01,
                    f"{v:.2f}",    ha='center',    va='bottom',    fontsize=7,
rotation=90)

plt.tight_layout()
plt.savefig('/content/EVAL7_metricas_por_clase.png',dpi=150,bbox_inches='ti
ght'); plt.show()
print("  Figura guardada: EVAL7_metricas_por_clase.png")

modelo_final = resultados[mejor_modelo]['modelo']
test_generator.reset()
batch_x, batch_y = next(test_generator)
preds = modelo_final.predict(batch_x, verbose=0)

fig, axes = plt.subplots(4, 4, figsize=(18,18))
fig.suptitle(f"PREDICCIONES CON {mejor_modelo}\n"
             "Verde=Correcto  Rojo=Incorrecto  (confianza en %)",
             fontsize=13, fontweight='bold')

for i in range(16):
    ax = axes[i//4, i%4]
    img = (batch_x[i]*255).astype(np.uint8)
    ax.imshow(img)
    pred_idx  = np.argmax(preds[i])
    true_idx  = np.argmax(batch_y[i])
    conf      = preds[i][pred_idx]*100
    pred_cls  = idx_to_class[pred_idx]
    true_cls  = idx_to_class[true_idx]
    correcto  = pred_idx == true_idx
    col       = '#2ecc71' if correcto else '#e74c3c'
    simbolo   = '' if correcto else ''
    ax.set_title(f"{simbolo}          Pred:          {pred_cls[:14]}\nReal:
{true_cls[:14]}\nConf: {conf:.1f}%",
                 fontsize=9, color=col, fontweight='bold')
    ax.axis('off')
    for spine in ax.spines.values():
        spine.set_edgecolor(col); spine.set_linewidth(3)
```

```
plt.tight_layout()
plt.savefig('/content/EVAL8_predicciones.png',dpi=150,bbox_inches='tight');
plt.show()
print("  Figura guardada: EVAL8_predicciones.png")
print("="*70)

print("\n" + "="*70)
print("  GUARDANDO MODELOS")
print("="*70)

for nombre, data in resultados.items():
    fn = f'/content/modelo_{nombre.lower().replace("-","_")}.h5'
    data['modelo'].save(fn)
    print(f"  Guardado: {fn}")

print("\n" + "="*70)
print("  RESUMEN FINAL DEL PROYECTO")
print("="*70)

print(f)
for nombre, m in evaluaciones.items():
    print(f"    • {nombre}:")
    print(f"                                    Accuracy={m['accuracy']*100:.2f}%
F1={m['f1_weighted']:.4f}  "
          f"AUC={m['auc_roc']:.4f}                    κ={m['cohen_kappa']:.4f}
MCC={m['mcc']:.4f}")
    print(f"        Tiempo entrenamiento: {m['tiempo_min']:.1f} min")

print(f)
print("="*70)
print("  PROYECTO COMPLETADO EXITOSAMENTE")
print("="*70)

def predecir_billete(imagen_entrada, modelo=None, top_k=5):

    if modelo is None:
        modelo = resultados[mejor_modelo]['modelo']

    if isinstance(imagen_entrada, str):
        img = cv2.cvtColor(cv2.imread(imagen_entrada), cv2.COLOR_BGR2RGB)
    else:
        img = imagen_entrada.copy()

    img_r = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    pred  = modelo.predict(img_r[np.newaxis]/255.0, verbose=0)[0]
    top   = np.argsort(pred)[::-1][:top_k]

    fig = plt.figure(figsize=(18,6))
    gs  = fig.add_gridspec(1,4)

    ax0 = fig.add_subplot(gs[0])
    ax0.imshow(img); ax0.axis('off')
    ax0.set_title(f"Predicción:                {idx_to_class[top[0]]}\nConf:
{pred[top[0]]*100:.1f}%",
                  fontsize=11,fontweight='bold',color='darkgreen')

    ax1 = fig.add_subplot(gs[1])
```

```
    ax1.imshow(PDI.clahe(img_r));                                ax1.axis('off');
ax1.set_title("CLAHE",fontsize=10)

    ax2 = fig.add_subplot(gs[2])
    ax2.imshow(PDI.canny(img_r), cmap='gray'); ax2.axis('off')
    ax2.set_title("Canny (bordes)", fontsize=10)

    ax3 = fig.add_subplot(gs[3])
    colores_bar = ['#2ecc71']+['#e74c3c']*(top_k-1)
    ax3.barh([idx_to_class[i][:20] for i in reversed(top)],
             [pred[i] for i in reversed(top)],
             color=list(reversed(colores_bar)), edgecolor='white')
    ax3.set_xlabel("Probabilidad",fontsize=10)
    ax3.set_title(f"Top-{top_k} clases",fontsize=11,fontweight='bold')
    ax3.set_xlim([0,1]); ax3.grid(axis='x',alpha=0.3)
    for spine in ax3.spines.values(): spine.set_visible(False)

    plt.suptitle("ANÁLISIS DE BILLETE - SISTEMA DE DETECCIÓN DE
FALSIFICACIONES",
                 fontsize=13,fontweight='bold',y=1.02)
    plt.tight_layout(); plt.show()

    return {
        'clase': idx_to_class[top[0]],
        'confianza': float(pred[top[0]]),
        'top_k': {idx_to_class[i]: float(pred[i]) for i in top}
    }

print("  Función predecir_billete() disponible")
print("  Uso: resultado = predecir_billete('/ruta/imagen.jpg')")
print("  O:   resultado = predecir_billete(imagen_array)")
print("="*70)
```