Modelo para la detección de imágenes generadas por IA

Instituto Tecnológico de Estudios Superiores de Occidente

(ITESO)

Roberto Garrido Hernández
Marcos Antonio Fierros Estrada
Rodrigo Emmanuel Macias Pantoja

29 de Noviembre del 2025

1 Introducción

El rápido avance de los modelos generativos ha incrementado la necesidad de desarrollar sistemas capaces de detectar si una imagen fue creada por inteligencia artificial (IA) o capturada del mundo real. La facilidad con la que estas imágenes pueden manipular información visual plantea retos en ámbitos como la verificación digital, la seguridad, el periodismo y la prevención de desinformación.

El presente trabajo tiene como objetivo desarrollar, entrenar y evaluar dos modelos distintos para la detección de imágenes generadas por IA con el fin de comparar sus desempeños. El primer enfoque emplea un modelo híbrido que combina representaciones profundas extraídas con una CNN (embeddings CNN) junto con características diseñadas manualmente (features handccrafted), las cuales se integran mediante un perceptrón multicapa (MLP). El segundo enfoque emplea un Vision Transformer (ViT) preentrenado, aprovechando técnicas modernas de transfer learning. La evaluación de los modelos se realizará utilizando y analizando métricas estándar de clasificación.

2 Marco Teórico

2.1 Detección de imágenes generadas por IA

La detección de imágenes sintéticas busca identificar patrones sutiles introducidos por modelos generativos. Aunque estas imágenes puedan parecer realistas al ojo humano, suelen contener artefactos estadísticos o estructurales que pueden ser detectados mediante técnicas de visión computacional. Los métodos de detección pueden utilizar características obtenidas manualmente, modelos de aprendizaje profundo o enfoques híbridos.

2.2 Handcrafted features

Las características diseñadas a mano consisten en descriptores creados manualmente basados en intuiciones o propiedades estadísticas de la imagen. Ejemplos comunes de características forenses incluyen Photo Response Non-Uniformity (PRNU), Local Binary Patterns (LBP), o métricas de ruido. En la detección de contenido sintético, estas características pueden capturar irregularidades que mejoren el desempeño que tendría un modelo profundo por sí solo. Su combinación con embeddings de CNN permite crear modelos híbridos que integran conocimiento explícito con aprendizaje profundo.

A continuación, se describen las características forenses que se utilizarán en el modelo híbrido:

- **LBP:**
  - Captura textura local
  - Es robusto a cambios de iluminación

- **Sobel[1]:**
  - Detecta bordes en la imagen mediante el cálculo de gradientes horizontales y verticales.
  - Permite identificar límites de objetos y estructuras, útiles para distinguir patrones sintéticos de reales.
- **Haralick/GLCM[2][3]:**
  - Analiza la textura de la imagen a través de la co-ocurrencia de intensidades de gris.
  - Sus características (contraste, homogeneidad, energía y correlación) ayudan a detectar texturas artificiales en imágenes generadas por IA.
- **Histograma[4]:**
  - Describe la distribución global de intensidades de píxeles en la imagen.
  - Permite identificar anomalías tonales y distribuciones uniformes típicas de imágenes sintéticas.
- **ELA:**
  - Detecta distribuciones irregulares de ruido.
  - Es usado para detectar ediciones en la imagen.
- **PRNU:**
  - Captura la sensibilidad de luz en los diferentes pixeles.
  - Es una característica debido a errores en manufactura.
  - Está presente en todos los chips de sensores de imagen.

2.3 Redes Neuronales Convolucionales (CNN)

Una CNN es un tipo de red neuronal diseñada para procesar datos con estructura espacial, como imágenes. Sus capas convolucionales aprenden filtros que detectan bordes, texturas y patrones complejos[5].

Las CNN suelen emplearse como extractores de características: en lugar de entrenarlas desde cero, se utilizan sus capas finales para obtener un embedding[6] o vector que representa el contenido visual. Por ejemplo, en la implementación se utiliza ResNet50, que procude un embedding de 2040 dimensiones. Este embedding puede luego combinarse con otros descriptores para entrenar un clasificador adicional.

El modelo seleccionado, ResNet50, es una Red Neuronal Convolucional (CNN) entrenada en un conjunto de aproximadamente 1 millón de imágenes y 1,000 clases diferentes (ImageNet-1k)[7].

2.4 Perceptrón Multicapa (MLP)

El MLP es una red neuronal compuesta por capas densas totalmente conectadas. En este trabajo se usa como clasificador final: recibe la concatenación del embedding de la CNN y el vector de características manuales y aprende a asignar cada imagen en una de dos clases: real o generada por IA.

2.5 Vision Transformer (ViT)

Los Vision Transformers son una arquitectura basada en transformers, originalmente diseñada para texto, pero adaptada para imágenes dividiéndolas en pequeños parches que funcionan como "tokens". El ViT aprende relaciones globales entre estos parches mediante mecanismos de atención[8].

El modelo seleccionado, google/vit-base-patch16-224-in21k, es un ViT entrenado en un conjunto de más de 14 millones de imágenes y 21,841 clases diferentes (ImageNet-21k)[9].

2.6 Transfer learning

El transfer learning consiste en reutilizar modelos ya entrenados en tareas generales para resolver tareas nuevas con menos datos y menor tiempo de entrenamiento. En visión computacional, esto es común con CNNs o ViTs preentrenados: se aprovecha el conocimiento previo del modelo (capacidad para detectar formas, texturas y estructuras) y solo se ajustan sus capas finales. Esto mejora el desempeño y evita tener que entrenar desde cero.

2.7 Full Fine-Tuning

El *Full Fine-Tuning* (ajuste fino completo) es la estrategia de *Transfer Learning* utilizada en este trabajo. Consiste en tomar un modelo pre-entrenado (como el ViT) y ajustar los pesos de todas sus capas (no solo las finales) durante el entrenamiento en la tarea específica de detección de imágenes sintéticas.

Se elige esta estrategia porque, si bien las capas iniciales de ViT ya han aprendido características generales, la detección de imágenes sintéticas a menudo depende de artefactos muy sutiles y de baja frecuencia. Permitir que los gradientes fluyan y modifiquen todas las capas hace que el modelo sea capaz de adaptar sus filtros iniciales específicamente para capturar estos artefactos específicos de los modelos generativos, logrando un mejor desempeño que si solo se ajustaran las capas finales.

2.8 Optimizador Adam

El *Optimizador Adam* (Adaptive Moment Estimation) es un algoritmo de optimización popular y eficiente utilizado para ajustar los pesos de la red neuronal durante el entrenamiento[10].

- Adam utiliza estimaciones del primer momento (la media de los gradientes) y del segundo momento (la varianza de los gradientes) de las pendientes. Esto permite que el algoritmo calcule tasas de aprendizaje adaptativas e individuales para cada parámetro, asegurando una convergencia rápida y estable.
- Adam es el optimizador por defecto en muchos modelos de aprendizaje profundo debido a su robustez y eficiencia en una amplia variedad de problemas, incluidas las tareas de visión con grandes modelos como ViT.

2.9 BCEWithLogitsLoss

BCEWithLogitsLoss (Binary Cross-Entropy with Logits Loss) es la función de pérdida utilizada en este trabajo, siendo la elección estándar para la clasificación binaria (Real vs. Generada por IA)[6].

Esta función combina dos pasos cruciales en una sola operación computacionalmente más estable:

1. Recibe los *logits* ($\hat{y}$), que son la salida directa, sin procesar, de la capa final de la red neuronal.
2. Aplica internamente la función de activación Sigmoid y la fórmula de la *BCE* (Binary Cross Entropy) contra la etiqueta real (y). Esto resulta en un cálculo numérico más estable que aplicar la Sigmoid y BCE por separado. Al operar directamente sobre los *logits*, evita problemas comunes como el *overflow* o *underflow* que pueden ocurrir al calcular logaritmos de valores cercanos a cero después de una Sigmoid.

2.10 Métricas de Evaluación

La evaluación del desempeño se realiza utilizando métricas específicas de clasificación que son cruciales para entender el rendimiento de un clasificador binario.

Recall (Sensibilidad)

El *Recall* mide la capacidad del modelo para identificar correctamente todas las instancias positivas reales. Se calcula como:

$$\text{Recall} = \frac{\text{Verdaderos Positivos (VP)}}{\text{Verdaderos Positivos (VP)} + \text{Falsos Negativos (FN)}}$$

En la detección de imágenes sintéticas, el *Recall* es fundamental. Un Falso Negativo (FN) significa que una imagen generada por IA fue clasificada erróneamente como real. Priorizar el *Recall* asegura que la mayoría de las imágenes fraudulentas sean detectadas, minimizando el riesgo en seguridad o desinformación.

F1-Score (Puntuación F1)

El *F1-Score* es la media armónica de la *Precision* y el *Recall*. Proporciona un equilibrio entre identificar correctamente las instancias positivas (Recall) y evitar clasificar erróneamente las instancias negativas como positivas (Precision).

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

Una puntuación F1 alta indica que el modelo es robusto, ya que esto no se podría lograr con una precisión baja y un *recall* alto (o viceversa). Es una métrica de resumen general que confirma que el modelo no está sobrecompensando en ninguna de las dos métricas.

ROC-AUC (Area Under the Receiver Operating Characteristic Curve)

El *ROC-AUC* mide la capacidad del modelo para distinguir entre clases. Es insensible al desequilibrio de clases y proporciona una visión integral del rendimiento del modelo, independientemente de dónde se establezca el umbral de decisión final. Una puntuación alta (cercana a 1.0) confirma la capacidad discriminativa intrínseca del modelo.

3 Metodología

3.1 Procesamiento de Datos

El desarrollo se enfoca en la preparación del conjunto de datos inicial para generar las representaciones que serán utilizadas por el modelo **MLP Híbrido**.

El procesamiento se centró en la uniformidad de las imágenes y la extracción del vector de características híbrido y del embedding de la CNN.

1. Preprocesamiento de Imágenes

Todas las imágenes del conjunto de datos fueron preprocesadas y normalizadas a una resolución fija de 224x224 píxeles. Adicionalmente, se obtuvo la versión en escala de grises para el cálculo de ciertas características forenses.

2. Extracción de Características Manuales (`handcrafted features`)

Para capturar artefactos y estadísticas sutiles introducidas por los modelos generativos, se implementó un extractor de características manuales. Este vector final de **58 dimensiones** se construye a partir de la concatenación de las siguientes técnicas forenses:

- **Sobel (Detección de Bordes)**:
    1. Se aplicaron filtros Sobel en las direcciones horizontal (G_x) y vertical (G_y) sobre la versión en escala de grises de la imagen.

2. Se calculó la magnitud del gradiente en cada píxel:

$$M(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

3. A partir de la magnitud se extrajeron estadísticas globales: la media y la desviación estándar, que reflejan la intensidad y variabilidad de los bordes.
4. Estas métricas permiten diferenciar imágenes reales (bordes afectados por ruido y condiciones físicas) de imágenes IA (bordes más uniformes y definidos).

- **Haralick/GLCM (Texturas)**:
  1. Se construyó la *Gray-Level Co-occurrence Matrix* (GLCM) considerando una distancia de 1 píxel y dirección horizontal.
  2. A partir de la GLCM se calcularon las siguientes propiedades:
     - **Contraste**: mide la variación local de intensidades.
     - **Homogeneidad**: evalúa la uniformidad de la textura.
     - **Energía**: refleja la repetitividad de patrones.
     - **Correlación**: mide la dependencia lineal entre intensidades vecinas.
  3. Estas características permiten identificar texturas naturales (con irregularidades propias de materiales y sensores) frente a texturas sintéticas (más uniformes o repetitivas).

- **Histograma (Distribución Global de Intensidades)**:
  1. Se calculó el histograma de intensidades en escala de grises, dividiendo el rango [0,255] en 32 *bins*.
  2. Cada bin representa la proporción de píxeles cuya intensidad cae en ese intervalo, normalizando la distribución:

$$h_i = \frac{n_i}{N}, \quad \sum_{i=1}^{32} h_i = 1$$

  3. El vector de frecuencias resultante describe la tonalidad global de la imagen.
  4. Imágenes reales presentan irregularidades en la distribución por condiciones de iluminación y ruido; imágenes IA tienden a mostrar distribuciones más uniformes o artificialmente balanceadas.

- **PRNU** (*Photo-Response Non-Uniformity*): Residuo de ruido intrínseco de la cámara, útil para identificar manipulaciones.

  El proceso que realiza el código es el siguiente:

1. Reducción de ruido mediante Transformada wavelet

   En esta etapa se aplica un filtro de wavelet denoising a la imagen. Esto con el objetivo de estimar la componente estructural "limpia" de la imagen. El sensor posee ruido propio, pero ese ruido queda principalmente en la imagen sin denoise. Por lo tanto, aplicar un suavizado wavelets obtenemos una aproximacion de contenido real sin ruido. Para utilizar la función de denoising se utilizaron los parámetros siguientes:

   - image. Imagen a la que se le aplicara el denoising.
   - mode (soft,hard). Es un parámetro opcional para mencionar cual es el tipo de denoising utilizado. La documentación comenta que utilizar "soft" regresa los mejores resultados. Por lo tanto, se utiliza el "soft".
   - method (BayesShrink, VIsuShrink). Son los métodos que se pueden utilizar para umbralización. Por default, se utilizó el BayesShrink.

2. Obtención del residual (ruido estimado del sensor)

   Recordando la siguiente ecuación:

   $$W = Im_{out} - denoise(Im_{out})$$

   Aquí se calcula la W de la ecuación. Restamos la imagen sin ruido de la imagen original. El resultado es el residual, que contiene principalmente ruido de luminancia, ruido aleatorio, y especialmente el componente PRNU, que es el ruido fijo del sensor. Este residual es una aproximación del ruido característico que la cámara imprime siempre en sus fotos.

3. Estimamos PRNU

   Recordando la siguiente ecuación:

   $$F = \frac{W}{Im_{in}}$$

   Finalmente, se divide el residual por la imagen para obtener una estimación del patrón PRNU. No se trata de una normalización, sino de la aplicación directa del modelo matemático que describe la relación entre el residual y el patrón de ruido del sensor.

- **ELA** (*Error Level Analysis*): Destaca diferencias en el nivel de compresión JPEG, indicando áreas de edición o falsificación.

  El proceso que se realiza es el siguiente:

  1. Conversión de formato de color

     La imagen cargada por la librería utilizada OpenCV está en formato BGR, por lo que primero se convierte a RGB para trabajar en un espacio de color estándar.

2. Recompresión controlada en JPEG

   La imagen se vuelve a comprimir artificialmente utilizando un nivel de calidad definido. Esta recompresión introduce errores predecibles.

3. Decodificación de la imagen recompresionada

   Se lee nuevamente la imagen JPEG generada, lo cual produce una versión recomprimida de la imagen original. Además, se convierte la salida de la decodificación a RGB para poder calcular la diferencia.

4. Cálculo de diferencias entre la imagen original y la recompresionada.

   En este paso conseguimos el ELA. Calculamos la diferencia absoluta pixel a pixel. Cuanto mayor sea la diferencia, mayor es el error introducido por la recompresión.

5. Conversión a escala de grises para visualización del ELA

   Finalmente, la imagen de diferencias se convierte a escala de grises, lo que simplifica la visualización de los niveles de error.

3. Extracción de Embeddings CNN (ResNet50)

En paralelo, se utilizó una CNN basada en el modelo ResNet50 pre-entrenado en ImageNet-1k para extraer un vector de **2048 dimensiones** (el embedding) por imagen. Este vector actúa como una representación de alto nivel del contenido visual.

4. Estandarización y Vector Híbrido

Los vectores de características manuales fueron estandarizados para asegurar que cada característica contribuya de manera equitativa al entrenamiento. Finalmente, las características manuales y el embedding CNN se concatenaron para formar el **vector de características híbrido** de 2106 dimensiones que alimenta al clasificador MLP.

3.2 División de Datos

La división de los datos se realizó para garantizar conjuntos de entrenamiento, prueba y validación distintos, esenciales para evaluar la generalización del modelo. El conjunto de datos inicial[11] consta de 10,000 imágenes con un balanceo de clases equilibrado (50% reales, 50% IA). Se empleó una división estratificada para asegurar que la proporción de imágenes reales y sintéticas fuera constante en todos los conjuntos.

- **División Principal (70% / 30%):** El conjunto de datos inicial se dividió en entrenamiento (70%) y un conjunto temporal (30%) manteniendo el equilibro de clases.
- **División Secundaria (15% / 15%):** El conjunto temporal (30%) se dividió equitativamente para obtener los conjuntos de prueba (Test, 15%) y validación (Validation, 15%).

Esta estrategia resultó en los siguientes conjuntos de datos finales, guardados en archivos `.csv` separados:

Datasets para el Modelo Híbrido (MLP Híbrido)

Estos conjuntos contienen el vector de características híbrido y sus etiquetas (real e IA):

- `hybrid_train.csv` (70%)
- `hybrid_test.csv` (15%)
- `hybrid_val.csv` (15%)

Datasets para el Modelo ViT

Dado que el modelo ViT realiza su propia extracción de características a partir de la imagen cruda, estos conjuntos solo contienen las rutas de las imágenes y las etiquetas, sin las características manuales ni los embeddings:

- `vit_train.csv` (70%)
- `vit_test.csv` (15%)
- `vit_val.csv` (15%)

3.3 Entrenamiento de los Modelos

El proceso de entrenamiento se ejecutó de forma independiente para el **Clasificador MLP Híbrido** y el **Vision Transformer (ViT)**, utilizando los conjuntos de datos de entrenamiento y prueba generados previamente.

Configuraciones Generales de Entrenamiento

- **Optimizador:** Se empleó **Adam** en ambos modelos, ajustando los pesos de la red de manera adaptativa.
- **Función de Pérdida: BCEWithLogitsLoss** fue la función de pérdida elegida para la clasificación binaria, garantizando estabilidad numérica al operar directamente sobre los *logits* de la capa de salida.
- **Criterio de mejor modelo:** Se guardaron los modelos con el mejor desempeño en la métrica *Recall* sobre el conjunto de prueba.

3.3.1 Entrenamiento del Clasificador MLP Híbrido

El modelo **MLP Híbrido** es una red de tres capas densas, diseñada para recibir como entrada la concatenación del embedding CNN y las características manuales, un **vector de características híbrido** de 2106 dimensiones.

- **Arquitectura:** La red sigue un diseño con dos capas ocultas o intermedias, aplicando regularización en cada una.
- **Hiperparámetros:**
  - **Neuronas primer capa oculta:** 256
  - **Neuronas segunda capa oculta** 64
  - **Épocas:** 50
  - **Tamaño del Lote (*Batch Size*):** 16
  - **Tasa de Aprendizaje (*Learning Rate*):** 1×10^{-4}

El entrenamiento gestionó la iteración manual sobre los lotes de datos, la propagación hacia adelante, el cálculo de la pérdida, la retropropagación y la actualización de pesos.

3.3.2 Entrenamiento del Vision Transformer (ViT)

El entrenamiento del ViT se basó en el proceso de **Full Fine-Tuning** del modelo pre-entrenado `google/vit-base-patch16-224-in21k`.

- **Procesamiento de Imágenes:** Se utiliza la extracción de características del propio modelo ViT, este componente se encarga de aplicar automáticamente el reescalado, la normalización y el formato tensorial adecuado a partir de la imagen cruda para la entrada al modelo ViT.
- **Full Fine-Tuning:** Se ajustaron los pesos de todas las capas del modelo ViT, incluyendo la gran mayoría de sus $\approx$ 86 millones de parámetros, permitiendo que la red se especialice en los sutiles patrones de las imágenes sintéticas.
- **Hiperparámetros:**
  - **Épocas:** 5
  - **Tamaño del Lote (*Batch Size*):** 16
  - **Tasa de Aprendizaje (*Learning Rate*):** 2×10^{-5}

4 Resultados

Se presenta el desempeño del **Modelo MLP Híbrido)** y el **Modelo ViT (google/vit-base-patch16-224-in21k)** sobre el conjunto de datos de validación en la tarea de detección de imágenes sintéticas utilizando métricas como el Recall, F1-Score y AUC.

4.1 Clasificador MLP Híbrido

El clasificador híbrido, que combina el *embedding* de la CNN ResNet50 con características manuales, mostró un excelente desempeño caracterizado por su alta velocidad.

Table 1: Evaluación del Clasificador MLP Híbrido

Métrica	**Valor**	**Interpretación**
Recall	0.9947	Identificó correctamente el 99.47% de las imágenes sintéticas.
F1-score	0.9664	Muy buen equilibrio entre *Precisión* y *Recall*.
AUC	0.9929	Capacidad discriminativa excelente, muy cercana a la perfección.
Tiempo de Predicción	0.0846 s	Extremadamente bajo, adecuado para inferencia en tiempo real con limitaciones en hardware.

Análisis de la Curva ROC AUC (MLP Híbrido)

La curva ROC para el clasificador híbrido (Figura 1) también demuestra una capacidad de clasificación excepcional con un AUC de 0.9929. La curva está fuertemente pegada a la esquina superior izquierda, confirmando su capacidad discriminativa robusta a través de todos los umbrales de decisión.

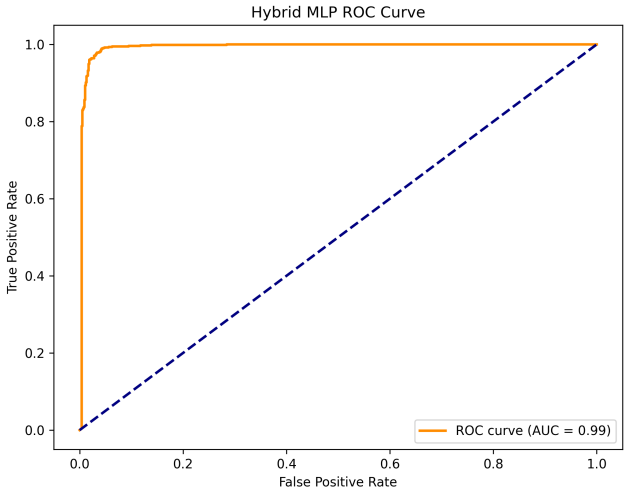

Figure 1: Curva ROC AUC del Clasificador MLP Híbrido.

4.2 Modelo Vision Transformer (ViT)

El modelo ViT, basado en la arquitectura de *transformers* pre-entrenada en ImageNet-21k, demostró un desempeño **cercano a la perfección** en las métricas de clasificación.

Table 2: Evaluación del Modelo ViT

Métrica	**Valor**	**Interpretación**
Recall	1.0000	Identificó correctamente el 100% de las imágenes sintéticas.
F1-score	0.9888	Excelente equilibrio entre *Precisión* y *Recall*.
AUC	1.0000	Capacidad discriminativa perfecta.
Tiempo de Predicción	16.5798 s	Alto costo computacional por la complejidad de la arquitectura de transformers.

Análisis de la Curva ROC AUC (ViT)

La curva ROC para el modelo ViT (Figura 2) confirma su capacidad discriminativa, alcanzando un Área Bajo la Curva (AUC) de 1.0000. La curva se eleva de manera inmediata, indicando una Tasa de Verdaderos Positivos del 100% con una Tasa de Falsos Positivos casi nula, demostrando una separación ideal entre las clases.

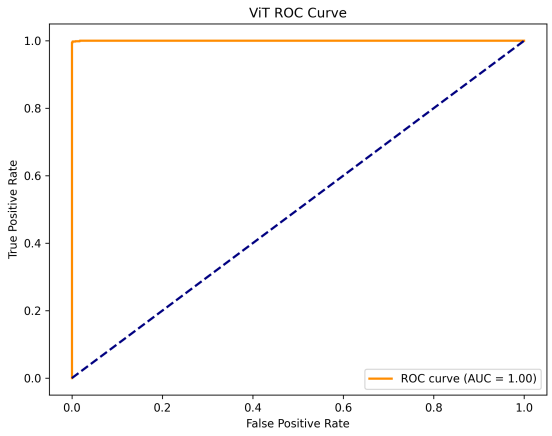

Figure 2: Curva ROC AUC del modelo Vision Transformer (ViT).

5 Conclusiones

La tabla 3 resume las métricas entre los dos enfoques evaluados.

Table 3: Comparación de Desempeño y Eficiencia

Característica	**ViT (Vision Transformer)**	**MLP Híbrido (CNN + Características)**
Efectividad Máxima (AUC)	**1.0000**	0.9929
Recall	**1.0000**	0.9947
Velocidad de Predicción	16.58 segundos	**0.08** segundos

El ViT demostró ser superior en precisión con una precisión absoluta (Recall de 1.0000), siendo ideal para aplicaciones críticas donde la minimización de Falsos Negativos es la prioridad absoluta. Sin embargo, el MLP Híbrido es **más de 200 veces más rápido** en el tiempo de predicción, ofreciendo una opción de alto rendimiento y eficiencia computacional. La alta efectividad de ambos modelos sugiere que, tanto los *features* extraídos por la arquitectura de transformers, como la combinación de descriptores convolucionales y manuales, son metodologías robustas para la detección de imágenes sintéticas.

Las características manuales (Sobel, Haralick/GLCM, Histogramas, LBP, PRNU y ELA) aportaron información complementaria que enriqueció la representación visual. Aunque algunas de ellas mostraron un poder discriminativo limitado de manera individual, su integración en un vector conjunto permitió capturar artefactos sutiles que los modelos profundos no detectan fácilmente. El modelo híbrido evidenció que la combinación de descriptores forenses con embeddings de CNN mejora la capacidad de detección en escenarios donde existen imágenes sintéticas. Esto confirma que las features manuales son útiles para reforzar la sensibilidad del sistema frente a irregularidades estadísticas y texturales.

Referencias

[1] Sobel Filter (s.f.). *ScienceDirect Topics*. `https://www.sciencedirect.com/topics/computer-science/sobel-filter`

[2] Löfstedt T, Brynolfsson P, Asklund T, Nyholm T, Garpebring A (2019). Gray-level invariant Haralick texture features. *PLoS ONE*, 14(2): e0212110. `https://doi.org/10.1371/journal.pone.0212110`

[3] scikit-image (s.f.). GLCM Texture Features Documentation. *scikit-image*. `https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_glcm.html`

[4] Nguyen Khanh Son (2023). Unlocking Image Enhancement: A Guide to Histogram Processing and Equalization with OpenCV. *Medium*. `https://medium.com/@khanhson0811/unlocking-image-`

enhancement-a-guide-to-histogram-processing-and-equalization-with-opencv-8db4477e6ba6

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324. https://ieeexplore.ieee.org/document/726791

[6] I. Goodfellow, Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org

[7] K. He, X. Zhang, S. Ren, and J. Sun (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778. https://doi.org/10.1109/CVPR.2016.90

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, et al. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2010.11929

[9] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248-255. https://doi.org/10.1109/CVPR.2009.5206848

[10] D. P. Kingma and J. Ba (2014). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1412.6980

[11] Sala, A. (2023). AI vs Human Generated Dataset. *Kaggle*. https://www.kaggle.com/datasets/alessandrasala79/ai-vs-human-generated-dataset

Anexos

Disponibilidad del Código

El código implementado y utilizado durante el desarrollo del presente trabajo se encuentra disponible en el siguiente repositorio oficial:

https://github.com/Marcois/ProyectoIDI2