



**Detección de falsificación en imágenes
utilizando enfoques de Aprendizaje Automático**

Autores:

Daniel Abad Fundora C411
Anabel Benítez González C411
Enzo Rojas D'Toste C411

danielabad774@gmail.com
abenitez@estudiantes.matcom.uh.cu
enzord2001@gmail.com

Septiembre 2024

Abstract

In this paper, we focus on image forgery detection, a key issue in today's digital world. With the rise of image editing tools, it is essential to identify alterations to ensure the authenticity of visual content. Our goal is to develop and compare different machine learning methods for detecting these forgeries, evaluating their accuracy and reliability. We will use both traditional image processing techniques and modern deep learning models.

Índice

1. Introducción	4
2. Estado del Arte	5
2.1. Métodos Basados en Características	5
2.1.1. Análisis de Metadatos	5
2.1.2. Análisis de Inconsistencias Visuales	5
2.2. Métodos Basados en Machine Learning	5
2.2.1. Redes Neuronales Convolucionales (CNN)	5
2.2.2. Autoencoders	5
2.2.3. Redes Generativas Antagónicas (GAN)	5
2.3. Técnicas Avanzadas	5
2.3.1. Transfer Learning	5
2.3.2. Análisis Forense de Imágenes	5
2.3.3. Detección Basada en Huellas Dactilares de Imagen	6
2.4. Combinación de técnicas	6
2.4.1. ManTra-Net: Manipulation Tracing Network for Detection and Localization of Image Manipulations [16]	6
2.4.2. DF-Net: The Digital Forensics Network for Image Forgery Detection [17]	6
2.4.3. FOrensic ContrASTive cLustering (FOCAL) [18]	6
2.5. Desafíos y Tendencias Futuras	6
2.5.1. Desafíos	6
2.5.2. Tendencias Futuras	7
3. Dataset utilizado	8
3.1. Características del CASIA V2	8
3.2. Uso en la Investigación	8
3.3. Aplicaciones	8
4. Extracción de Características y Entrenamiento de un Modelo SVM	9
4.1. Carga de Imágenes	9
4.2. Extracción de Características	9
4.3. Preparación del Conjunto de Datos	9
4.4. Entrenamiento del Modelo SVM	9
4.5. Resultados	9
5. Utilizando técnicas básicas de Deep Learning	11
5.1. Utilizando U-Nets	11
5.1.1. Etapa de Contracción	11
5.1.2. Etapa de Expansión	11
5.1.3. Skip Connections	11
5.1.4. Resultados	12
5.2. Utilizando una Red Generativa Adversarial	13
5.2.1. Configuración del Generador	13
5.2.2. Configuración del Discriminador	13
5.2.3. Resultados	14
6. Utilizando técnicas de segmentación de imágenes y una CNN	15
6.1. ¿Qué es la compresión JPEG?	15
6.2. Enfoque seguido	16
6.3. Modelo Utilizado	17
6.4. Análisis de Resultados	17

7. Utilizando Transfer Learning	18
7.1. Extracción de Características	18
7.2. Ajuste Fino (Fine-Tuning)	18
7.3. Modelo importado	18
7.4. Resultados obtenidos	19
8. Evaluación de algunos métodos no supervisados para el análisis de imágenes	21
8.1. Utilizando Residual de Interpolación del Canal Verde CFA	21
8.1.1. Extracción del Canal Verde	21
8.1.2. Cálculo del Residual de Interpolación CFA	21
8.1.3. Cálculo de la Energía del Residual	21
8.1.4. Normalización de la Energía del Residual	21
8.1.5. Detección de Manipulación	22
8.1.6. Resultados	22
8.2. Utilizando el análisis del Ruido PRNU	23
8.2.1. Descripción del Algoritmo	23
8.2.2. Extracción del Ruido PRNU	23
8.2.3. Detección de Manipulación	23
8.2.4. Superposición de la Máscara sobre la Imagen Original	24
8.2.5. Resultados	24
9. Análisis de resultados	26
9.1. Repercusión ética de las soluciones	26
10. Conclusiones	27
11. Propuestas para futuros trabajos	28

1 Introducción

En el presente trabajo, abordamos un problema de gran relevancia en la era digital: la detección de imágenes editadas. Con el auge de las herramientas de edición de imágenes y la facilidad con la que se pueden modificar fotografías, identificar si una imagen ha sido alterada se ha convertido en una tarea crucial para asegurar la integridad y autenticidad de los contenidos visuales.

Detectar la manipulación en imágenes no es una tarea trivial, ya que implica reconocer alteraciones que pueden variar desde simples ajustes de color y brillo hasta complejas manipulaciones de objetos y personas dentro de la imagen.

Nuestro objetivo en este trabajo es desarrollar y comparar varios enfoques de Machine Learning para detectar estas modificaciones, evaluando cuál de ellos proporciona los resultados más precisos y robustos. Para lograr este objetivo, implementaremos y evaluaremos una serie de métodos que incluirán técnicas de procesamiento de imágenes tradicionales así como modelos avanzados de aprendizaje profundo (Deep Learning). Entre los enfoques considerados se encuentran la detección de inconsistencias en las características visuales, análisis de metadatos, uso de redes neuronales convolucionales (CNN) y técnicas de transferencia de aprendizaje.

Este trabajo representa no solo un desafío técnico significativo, sino también una oportunidad para aplicar los conocimientos adquiridos a lo largo de nuestra formación como Científicos de la Computación. Asimismo, buscamos contribuir con una solución a un problema relevante en la actualidad, asegurando la veracidad y confiabilidad de las imágenes en un mundo cada vez más digitalizado.

2 Estado del Arte

La detección de imágenes editadas es un campo de estudio activo y en evolución dentro de la visión por computadora y el procesamiento de imágenes. A continuación, presentamos un resumen de los enfoques más relevantes y avanzados en la literatura actual:

2.1 Métodos Basados en Características

2.1.1 Análisis de Metadatos

Los metadatos de las imágenes, como la información EXIF, pueden proporcionar pistas sobre si una imagen ha sido editada. Sin embargo, los editores de imágenes pueden modificar o eliminar estos datos, lo que limita la efectividad de este enfoque.

2.1.2 Análisis de Inconsistencias Visuales

Este enfoque se basa en identificar inconsistencias en las propiedades visuales de una imagen, como los bordes, las sombras y las texturas. Técnicas como el análisis de histograma y la detección de bordes se utilizan para encontrar discrepancias que podrían indicar manipulación.

2.2 Métodos Basados en Machine Learning

2.2.1 Redes Neuronales Convolucionales (CNN)

Las CNN son modelos de aprendizaje profundo que han demostrado ser altamente efectivos para tareas de clasificación y detección en imágenes. Para la detección de imágenes editadas, las CNN pueden entrenarse para reconocer patrones y características que son indicativos de manipulación. Ejemplos de arquitecturas populares incluyen VGG, ResNet y EfficientNet.

2.2.2 Autoencoders

Los autoencoders son un tipo de red neuronal utilizada para la reducción de dimensionalidad y la detección de anomalías. En el contexto de la detección de imágenes editadas, un autoencoder puede ser entrenado para reconstruir imágenes no editadas. Las diferencias significativas entre la imagen original y la reconstruida pueden señalar la presencia de ediciones.

2.2.3 Redes Generativas Antagónicas (GAN)

Las GAN, compuestas por un generador y un discriminador, se utilizan tanto para generar como para detectar imágenes falsas. En la detección de imágenes editadas, el discriminador se entrena para distinguir entre imágenes reales y aquellas generadas o manipuladas por el generador. Este enfoque es particularmente efectivo para detectar ediciones sutiles.

2.3 Técnicas Avanzadas

2.3.1 Transfer Learning

El aprendizaje por transferencia implica utilizar un modelo preentrenado en un conjunto de datos grande y luego afinarlo para una tarea específica, como la detección de imágenes editadas. Este método puede mejorar la precisión y reducir el tiempo de entrenamiento.

2.3.2 Análisis Forense de Imágenes

Las técnicas forenses analizan los artefactos introducidos durante el proceso de edición, como la interpolación de píxeles y la compresión JPEG. Estas técnicas pueden ser combinadas con algoritmos de machine learning para mejorar la detección.

2.3.3 Detección Basada en Huellas Dactilares de Imagen

Cada dispositivo de captura de imágenes (como cámaras y teléfonos móviles) deja una huella dactilar única debido a imperfecciones del sensor. Comparar estas huellas dactilares puede ayudar a identificar si una imagen ha sido manipulada o proviene de diferentes fuentes.

2.4 Combinación de técnicas

2.4.1 ManTra-Net: Manipulation Tracing Network for Detection and Localization of Image Manipulations [16]

- Se utiliza una red neuronal convolucional (CNN) para extraer características ricas y representativas directamente de los píxeles de la imagen.
- Las características extraídas por la CNN se procesan utilizando una red de memoria a largo corto plazo (LSTM) para identificar y trazar secuencias de manipulaciones en la imagen. Las características se organizan en secuencias que el LSTM procesa para detectar patrones que sugieren modificaciones en la imagen.
- Se generan mapas que asignan una probabilidad a cada píxel de la imagen original, indicando la probabilidad de que dicho píxel haya sido manipulado. Se aplican técnicas de suavizado y refinamiento para mejorar la precisión y reducir el ruido en los mapas de manipulación.

2.4.2 DF-Net: The Digital Forensics Network for Image Forgery Detection [17]

- DF-Net utiliza dos sub-redes, M1 y M2, ambas basadas en la arquitectura U-Net.
- Codificación y Decodificación: Empieza con una fase de codificación que reduce las dimensiones espaciales y aumenta las dimensiones de características. Luego, sigue una fase de decodificación que reconstruye la imagen a partir de las características aprendidas.
- Emplea conexiones de salto que permiten que algunas características aprendidas en la codificación se utilicen directamente en la decodificación, lo que ayuda a preservar los detalles de la imagen.
- M1 se entrena primero en todos los tipos de manipulaciones y luego se afina con forjados de tipo "copy-move". M2 se entrena exclusivamente con imágenes de "splicing".
- Se combinan las sub-redes M1 y M2, donde se toma el valor máximo de predicción de ambos para cada píxel.

2.4.3 FOrensic ContrAptive cLustering (FOCAL) [18]

- Combina el aprendizaje contrastivo a nivel de píxeles y la agrupación no supervisada.
- Entrenamiento mediante Aprendizaje Contrastivo: FOCAL supervisa la extracción de características usando máscaras de falsificación como guía para discriminación a nivel de píxeles.
- Fase de Pruebas con Agrupación No Supervisada: En lugar de utilizar un clasificador entrenado, FOCAL aplica un algoritmo de clustering en tiempo real (HDBSCAN) para asignar características a categorías de falsificado o auténtico basándose en la predominancia en la imagen, asumiendo que los píxeles falsificados son menos frecuentes.

2.5 Desafíos y Tendencias Futuras

2.5.1 Desafíos

- **Diversidad de ediciones:** Las ediciones pueden variar significativamente en complejidad y sutileza, desde ajustes menores de color hasta manipulaciones profundas.
- **Calidad y tamaño de datos:** La disponibilidad de conjuntos de datos etiquetados y de alta calidad es crucial para entrenar modelos efectivos.

- **Robustez a nuevas técnicas de edición:** Los métodos de detección deben adaptarse continuamente a las nuevas técnicas de edición que surgen.

2.5.2 Tendencias Futuras

- **Modelos híbridos:** La combinación de múltiples enfoques, como el análisis de características y técnicas de deep learning, puede ofrecer soluciones más robustas.
- **Aumento de datos sintéticos:** Generar datos sintéticos para entrenar modelos puede ayudar a abordar la escasez de datos etiquetados.
- **Mejora en la interpretabilidad:** Desarrollar modelos que no solo detecten ediciones, sino que también proporcionen explicaciones claras de cómo y dónde se realizó la manipulación.

En resumen, la detección de imágenes editadas es un campo dinámico que combina técnicas de procesamiento de imágenes tradicionales con métodos avanzados de machine learning. A medida que la tecnología avanza, es crucial continuar investigando y desarrollando enfoques innovadores para mantenerse a la vanguardia en la detección de manipulaciones de imágenes.

3 Dataset utilizado

El dataset **CASIA V2** (CASIA Image Tampering Detection Evaluation Database) es un conjunto de datos ampliamente utilizado para la investigación en la detección de manipulación de imágenes. Fue creado por el Instituto de Automatización de la Academia China de Ciencias (CASIA) y se ha convertido en una referencia en el campo debido a su diversidad y volumen de imágenes manipuladas.

3.1 Características del CASIA V2

- **Variedad de Manipulaciones:** El dataset incluye imágenes con diferentes tipos de manipulaciones, como copiar y pegar, recortar, y alterar partes de la imagen. Esto permite a los investigadores probar la efectividad de sus algoritmos en diversas situaciones.
- **Volumen de Imágenes:** CASIA V2 contiene un total de 12,614 imágenes, de las cuales 7,491 son imágenes auténticas y 5,123 son imágenes manipuladas. Esta gran cantidad de datos proporciona una base sólida para entrenar y evaluar modelos de detección de manipulación.
- **Anotaciones Detalladas:** Las imágenes manipuladas vienen con anotaciones que especifican las áreas que han sido alteradas. Estas anotaciones son cruciales para la evaluación precisa de los algoritmos de detección.
- **Diversidad de Escenarios:** El dataset incluye una amplia gama de escenarios y contextos, lo que asegura que los modelos entrenados con CASIA V2 sean robustos y puedan generalizar bien a diferentes tipos de imágenes.

3.2 Uso en la Investigación

CASIA V2 ha sido utilizado en numerosos estudios y publicaciones científicas para el desarrollo y la evaluación de métodos de detección de manipulación de imágenes. Al ser un dataset bien establecido y de acceso público, facilita la comparación de diferentes enfoques y algoritmos.

3.3 Aplicaciones

- **Entrenamiento de Modelos:** Las imágenes y anotaciones proporcionadas por CASIA V2 son ideales para entrenar modelos de aprendizaje profundo, como U-Nets, para detectar áreas manipuladas en las imágenes.
- **Evaluación de Algoritmos:** Las anotaciones detalladas permiten una evaluación precisa de los algoritmos, ayudando a medir métricas como la precisión, *recall* y *F1-score*.
- **Benchmarking:** CASIA V2 sirve como un estándar de referencia para comparar la efectividad de diferentes métodos y técnicas de detección de manipulación.

En resumen, CASIA V2 es un recurso valioso para la comunidad de investigadores en el campo de la detección de manipulación de imágenes, proporcionando una base sólida para el desarrollo y la evaluación de nuevos métodos y algoritmos.

4 Extracción de Características y Entrenamiento de un Modelo SVM

En este capítulo, se describe el proceso de extracción de características de imágenes y el entrenamiento de un modelo de Máquina de Vectores de Soporte (SVM) para la detección de imágenes manipuladas. Este proceso incluye la carga de imágenes, la extracción de varias características de cada imagen y el uso de estas características para entrenar y evaluar un modelo SVM.

4.1 Carga de Imágenes

Para el procesamiento de imágenes, se utiliza la función `cv2.imread` de OpenCV, que permite cargar imágenes en escala de grises. Esto simplifica el procesamiento y la extracción de características al trabajar con un único canal de intensidad.

4.2 Extracción de Características

Se extraen diversas características de las imágenes, cada una proporcionando información útil para la detección de manipulación:

- **Transformada de Fourier:** La transformada de Fourier se utiliza para obtener el espectro de magnitud de la imagen, revelando las frecuencias presentes. Esto es útil para identificar patrones que podrían indicar manipulaciones.
- **Características de Ruido:** Se calculan la media y la desviación estándar del ruido en la imagen. Estas características ayudan a detectar irregularidades que pueden ser indicativas de manipulación.
- **Detección de Bordes:** La detección de bordes se realiza utilizando el algoritmo de Canny. Los bordes pueden proporcionar información crucial sobre las discontinuidades en la imagen, que a menudo son resultado de manipulaciones.
- **Características de Textura** Se utilizan matrices de co-ocurrencia de niveles de gris (GLCM) para calcular características de textura, específicamente el contraste. La textura de una imagen puede cambiar notablemente cuando se manipulan ciertas áreas.
- **Segmentación de Imágenes** La segmentación se lleva a cabo mediante el algoritmo de k-means, dividiendo la imagen en k clústeres. Esto permite analizar diferentes regiones de la imagen por separado, lo cual es útil para identificar áreas manipuladas.

4.3 Preparación del Conjunto de Datos

Se itera sobre una colección de imágenes, extrayendo las características mencionadas anteriormente de cada una. Estas características se almacenan junto con etiquetas que indican si la imagen ha sido manipulada o no.

4.4 Entrenamiento del Modelo SVM

Una vez que se han extraído las características y preparado el conjunto de datos, se divide este conjunto en datos de entrenamiento y prueba. El modelo SVM se entrena utilizando los datos de entrenamiento. El SVM es un algoritmo de clasificación supervisada eficiente para este tipo de problemas debido a su capacidad para manejar grandes dimensiones de características y encontrar un hiperplano óptimo que separa las clases.

4.5 Resultados

Selección del Conjunto de Datos y Enfoque del Análisis

Debido a las limitaciones de memoria RAM y la carga computacional que implica almacenar tanto las fotos originales como los vectores de características extraídos de ellas, se optó por seleccionar un conjunto de datos reducido para el entrenamiento del modelo SVM. Los análisis subsiguientes se realizaron utilizando un subconjunto específicamente elegido por devolver los mejores resultados preliminares. Para las pruebas, se utilizó una selección aleatoria reducida del conjunto de prueba. Este enfoque está diseñado para evaluar más el

comportamiento del algoritmo al variar sus características, que para alcanzar las máximas precisiones posibles, ya que estas últimas pueden variar significativamente dependiendo del tamaño y la naturaleza del conjunto de entrenamiento utilizado. Este análisis permite entender mejor cómo las diferentes configuraciones afectan el rendimiento del algoritmo en un contexto controlado.

Precisión General del Modelo

Inicialmente, se entrenó el modelo SVM utilizando todas las características mencionadas. La precisión obtenida en el conjunto de prueba reducido fue del 72 %. Esto establece un punto de referencia para comparar el impacto de las características individuales.

Análisis de la Importancia de las Características

Para investigar qué características tenían más impacto en la precisión del modelo, se realizaron pruebas eliminando una característica a la vez. La tabla a continuación resume cómo la eliminación de cada característica afectó la precisión del modelo:

Característica Removida	Precisión (%)
Ninguna (base)	72
Transformada de Fourier	70
Ruido	72
Detección de Bordes	77
Texturas	72
Segmentación	58

Cuadro 1: Impacto de la eliminación de características individuales en la precisión del modelo SVM.

Pruebas Adicionales

Además, se realizó una prueba adicional utilizando solo las características de la transformada de Fourier y la segmentación, que demostraron tener el mayor impacto en la precisión. Este modelo simplificado mantuvo una precisión del 77 %, lo cual es notable considerando la reducción en la dimensionalidad. Es importante destacar que la característica de segmentación contribuye con la mayor cantidad de dimensiones al vector de características, implicando así una carga computacional significativa.

Conclusiones del Análisis de Características

Los resultados sugieren que la transformada de Fourier y la segmentación son críticas para la precisión del modelo, mientras que las otras características no tienen un impacto significativo o incluso pueden empeorar el rendimiento. Esta información es crucial para optimizar la eficiencia del modelo reduciendo la dimensionalidad sin comprometer la efectividad.

5 Utilizando técnicas básicas de Deep Learning

5.1 Utilizando U-Nets

Las U-Nets son una clase de redes neuronales convolucionales diseñadas originalmente para tareas de segmentación de imágenes biomédicas. Introducidas por Olaf Ronneberger et al. en 2015, su arquitectura se caracteriza por una estructura en forma de "U", que consta de una etapa de contracción y una etapa de expansión. Se pueden encontrar ejemplos de uso de las UNETs para clasificación de imágenes en [7], [8] y [9].

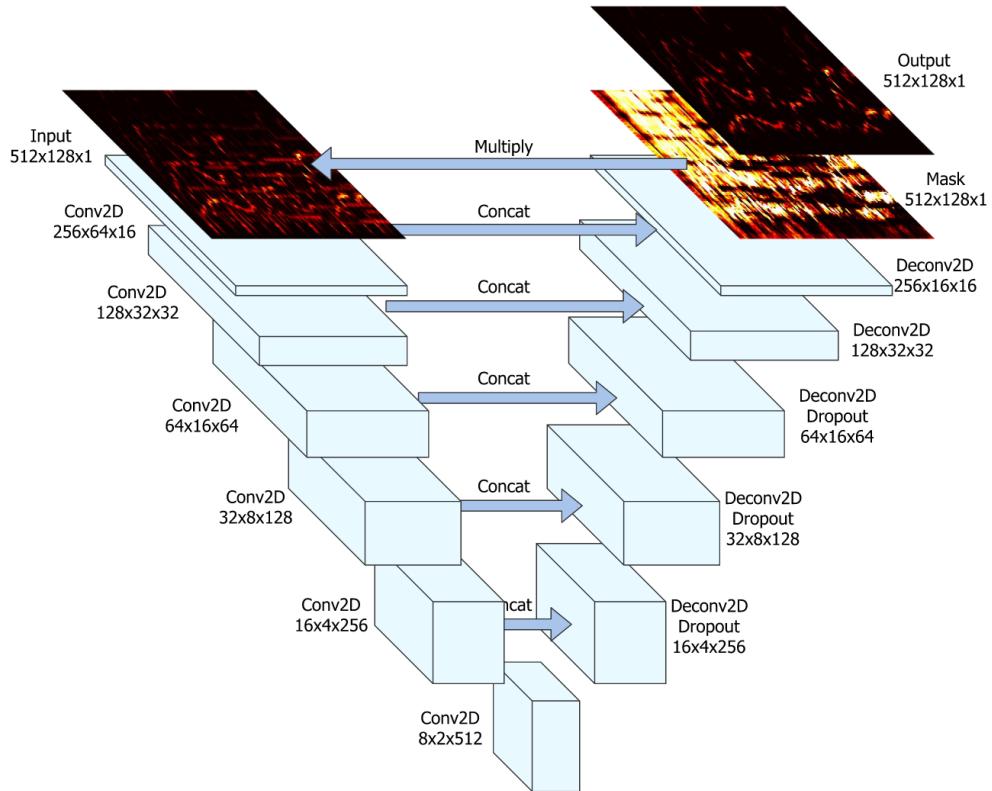


Figura 1: Arquitectura de una U-Net típica.

5.1.1 Etapa de Contracción

La etapa de contracción es similar a una CNN tradicional y consiste en varias capas de convolución seguidas de capas de agrupamiento (pooling). Durante esta etapa, se reduce la resolución espacial de la imagen mientras se incrementa el número de características detectadas.

5.1.2 Etapa de Expansión

La etapa de expansión es donde la resolución espacial se recupera mediante operaciones de upsampling y convolución. Además, las características aprendidas en la etapa de contracción se combinan con las características recuperadas, proporcionando una información más rica y detallada.

5.1.3 Skip Connections

Una característica clave de las U-Nets son las *skip connections*, que permiten la transferencia directa de características de la etapa de contracción a la etapa de expansión. Esto ayuda a preservar información detallada y mejora la precisión de la segmentación.

5.1.4 Resultados

El modelo presenta una velocidad de entrenamiento insuficiente, lo que limitó las pruebas a conjuntos de datos reducidos. Esta restricción resultó en un claro problema de overfitting, donde el modelo se ajustó excesivamente a los datos de entrenamiento, perdiendo capacidad de generalización. Los experimentos realizados no produjeron resultados satisfactorios, quedando por debajo de las expectativas para aplicaciones prácticas. El rendimiento observado indica que el modelo, en su estado actual, requiere una revisión significativa de su arquitectura y métodos de entrenamiento para superar estas limitaciones y alcanzar un desempeño aceptable.

5.2 Utilizando una Red Generativa Adversarial

Se configuró una GAN (Generative adversarial network) donde el generador se entrenó para crear imágenes que parecieran lo más reales posible, mientras que el discriminador aprendía a identificar estas como falsas. El objetivo era que el discriminador mejorara su capacidad para detectar manipulaciones en imágenes ajenas al conjunto de entrenamiento.

5.2.1 Configuración del Generador

El generador de la red GAN fue diseñado para transformar un espacio latente en imágenes RGB de alta resolución. La arquitectura del generador se implementó utilizando el modelo secuencial de Keras. La descripción detallada de la arquitectura es la siguiente:

- Comenzamos con una capa densa que recibe como entrada la dimensión latente del espacio. Esta capa se expande a una salida de 128 canales para cada pixel en una matriz de 38×38 , seguido de una activación ReLU.
- La salida se reorganiza mediante una capa de redimensionamiento a la forma $(38, 38, 128)$.
- Se aplica normalización por lotes para estabilizar el aprendizaje y mejorar la convergencia del modelo.
- Una serie de capas convolucionales transpuestas se utiliza para escalar la imagen de 38×38 a 76×76 y luego a 152×152 , usando cada vez una activación ReLU y normalización por lotes.
- La capa final es una convolución transpuesta que ajusta la salida a 150×150 píxeles y reduce la profundidad a 3 canales, correspondientes a una imagen RGB, utilizando una activación sigmoidal para normalizar los valores de los píxeles entre 0 y 1.
- Un recorte mediante *Cropping2D* se aplica para ajustar las dimensiones finales exactamente a $150 \times 150 \times 3$, recortando los bordes de la imagen generada para obtener dimensiones precisas.

Esta configuración del generador permite crear imágenes que, aunque inicialmente carecen de contexto real, se refinan gradualmente a través del entrenamiento adversario con el discriminador para simular imágenes reales lo más fielmente posible.

5.2.2 Configuración del Discriminador

El discriminador fue construido utilizando MobileNetV2, un modelo preentrenado proveniente de las aplicaciones de Keras, adaptado para la tarea específica de clasificar imágenes como reales o manipuladas. A continuación se detalla la configuración utilizada:

- El modelo base, MobileNetV2, se configuró para recibir imágenes del tamaño de la entrada especificada, sin incluir la capa superior y utilizando los pesos preentrenados de ImageNet. Esto proporciona una base sólida de características visuales aprendidas para la detección.
- Se activó la capacidad de entrenamiento del modelo base para permitir ajustes finos. Específicamente, se congelaron las capas anteriores a la capa número 120, permitiendo que las capas subsiguientes se ajustaran durante el entrenamiento para especializarse en la detección de manipulaciones.
- Se incorporó un módulo de aumento de datos directamente en el modelo, que realiza aleatoriamente volteos horizontales y rotaciones de hasta 0.2 radianes, para mejorar la robustez del discriminador frente a variaciones naturales en las imágenes de entrada.
- Las imágenes de entrada se preprocesan utilizando la función específica de MobileNetV2, que ajusta los valores de los píxeles para el rango que el modelo base espera.
- Posterior al procesamiento por el modelo base, se añade una capa de agrupación promedio global para reducir la dimensionalidad de los mapas de características, seguido de una capa densa que produce la salida final del modelo, un único valor que indica la probabilidad de que la imagen sea manipulada.

- El modelo se compiló con una función de pérdida de entropía cruzada binaria con logits, utilizando un optimizador RMSprop con una tasa de aprendizaje muy baja (0.00001), para realizar ajustes finos cuidadosos en el entrenamiento.

Esta configuración del discriminador permite no solo adaptar un modelo preentrenado para la tarea específica de identificación de manipulaciones, sino también ajustar su sensibilidad a través del entrenamiento supervisado directamente en nuestro conjunto de datos de imágenes manipuladas y no manipuladas.

5.2.3 Resultados

A continuación, se presentan algunos ejemplos de las imágenes generadas durante las distintas etapas del entrenamiento, ilustrando la evolución del generador.

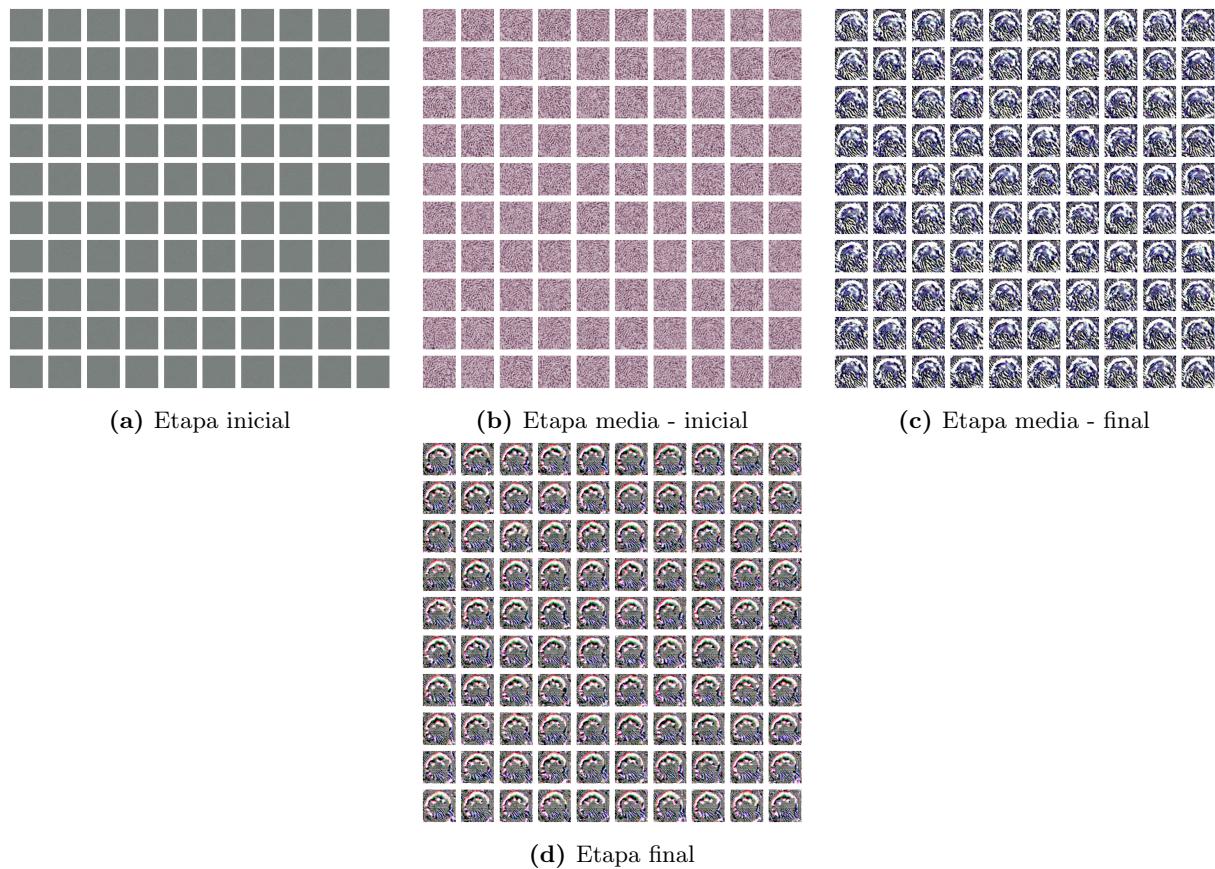


Figura 2: Evolución de las imágenes generadas por el generador en diferentes etapas del entrenamiento.

A pesar de observar patrones en las imágenes generadas, estos no contribuyeron a una mejora significativa en la detección de manipulaciones por parte del discriminador. Esto puede sugerir limitaciones inherentes al enfoque o la necesidad de ajustes en la arquitectura y el entrenamiento.

6 Utilizando técnicas de segmentación de imágenes y una CNN

Basados en los trabajos de [1], utilizamos una técnica que combina la recompresión de imágenes con Redes Neuronales Convolucionales (CNNs) para detectar imágenes falsificadas.

6.1 ¿Qué es la compresión JPEG?

La compresión JPEG (Joint Photographic Experts Group) es un estándar de compresión de imágenes ampliamente utilizado, especialmente para fotografías digitales. Este método combina técnicas de compresión con y sin pérdida para reducir el tamaño de los archivos, manteniendo una calidad visual aceptable. Fue introducido oficialmente en 1992 por el grupo del mismo nombre, un comité de trabajo que es una colaboración entre la ISO (International Organization for Standardization) y la IEC (International Electrotechnical Commission). A continuación, se un resumen del proceso y el trasfondo matemático de la compresión JPEG de [6].

Conversión de Espacio de Color

El proceso comienza con la conversión de la imagen del espacio de color RGB (Rojo, Verde, Azul) al espacio de color YCbCr. Donde:

- **Y** representa la luminancia (brillo).
- **C_b** y **C_r** representan la crominancia (diferencia de color respecto a la luminancia, para el azul y el rojo, respectivamente).

Este paso es crucial porque el ojo humano es más sensible a los cambios en la luminancia que a los cambios en la crominancia. Por lo tanto, se puede comprimir más los componentes de crominancia sin afectar mucho la calidad percibida.

Submuestreo de Crominancia

Dado que la percepción humana es menos sensible a los detalles de color, JPEG usa un proceso llamado submuestreo de crominancia para reducir la cantidad de datos de C_b y C_r. Un formato común es 4:2:0, donde la resolución de las componentes de color se reduce a la mitad en ambas dimensiones.

División en Bloques

La imagen se divide en bloques de 8×8 píxeles. Esta división permite aplicar compresión en pequeños segmentos de la imagen, lo cual es más manejable y eficiente en términos de procesamiento.

Transformada Discreta de Coseno (DCT)

Cada bloque de 8×8 se transforma usando la DCT. La DCT convierte los valores de píxeles del dominio espacial al dominio de frecuencia. Esto significa que los valores de los píxeles, que representan variaciones en la imagen, se descomponen en una suma de funciones de coseno con diferentes frecuencias y amplitudes.

Matemáticamente, para un bloque de 8×8 , la DCT se define como:

$$F(u, v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

donde:

- $f(x, y)$ es el valor de luminancia del píxel en la posición (x, y) en el bloque.
- $F(u, v)$ es el coeficiente DCT resultante.
- $C(u), C(v)$ son factores de normalización ($\frac{1}{\sqrt{2}}$ para $u, v = 0$ y 1 para los demás).

Cuantización

Después de la DCT, se aplica un proceso de cuantización. La cuantización reduce la precisión de los coeficientes DCT, eliminando las frecuencias menos perceptibles (principalmente las altas frecuencias que corresponden a cambios rápidos en la imagen). Cada coeficiente se divide por un valor de una tabla de cuantización y luego se redondea al número entero más cercano.

$$Q(u, v) = \text{round} \left(\frac{F(u, v)}{Q_{\text{table}}(u, v)} \right)$$

donde $Q_{\text{table}}(u, v)$ es un valor de la tabla de cuantización que puede ajustarse según el nivel de compresión deseado (calidad de imagen).

Codificación

Los coeficientes cuantizados se organizan en un recorrido en zigzag para agrupar los ceros. Luego, los coeficientes se codifican utilizando dos métodos:

- **Codificación de longitud de run:** Para los ceros consecutivos.
- **Codificación de Huffman:** Para los coeficientes no nulos, utilizando un código más corto para los valores más frecuentes.

Construcción del Archivo JPEG

Finalmente, la información comprimida se empaqueta en un archivo JPEG, que incluye los datos de los coeficientes codificados, junto con información sobre la tabla de cuantización y otras configuraciones.

6.2 Enfoque seguido

Los puntos clave de la técnica son:

- Las CNNs se inspiran en el sistema visual humano y son efectivas para tareas de visión por computadora.
- La falsificación de imágenes deja artefactos sutiles que las CNNs pueden detectar, aunque sean imperceptibles para el ojo humano.
- Cuando se recomprime una imagen falsificada, la parte alterada se comprime de manera diferente al resto debido a su origen distinto.
- El método propuesto:
 1. Toma una imagen potencialmente falsificada (A)
 2. La recomprime ($A_{\text{recompressed}}$)
 3. Calcula la diferencia entre A y $A_{\text{recompressed}}$ (A_{diff})
 4. En A_{diff} , la parte falsificada se destaca
 5. Usa A_{diff} como entrada para entrenar una CNN
- La CNN aprende a clasificar imágenes como auténticas o falsificadas basándose en estos patrones de diferencia de compresión.
- Se aprovecha el hecho de que las regiones falsificadas tienen una distribución estadística diferente de coeficientes DCT (Discrete Cosine Transform) comparadas con las regiones originales.

Enfoques similares han sido utilizados en [11], [12], [13] y [14].

6.3 Modelo Utilizado

El modelo consiste en una red neuronal convolucional (CNN) diseñada para detectar manipulación en imágenes, utilizando Keras con TensorFlow como backend. A continuación se describen las capas de nuestra CNN:

1. **Input:** Imágenes en escala de grises (1 canal) con resolución de 128x128 píxeles.
2. **Conv2D:** Tres capas convolucionales con 32 filtros y kernel de 3x3, activación ReLU.
3. **MaxPooling2D:** Capa de max-pooling con tamaño de pool de 2x2.
4. **Flatten:** Aplana las características en un vector unidimensional.
5. **Dense:** Capa densa con 256 neuronas y activación ReLU.
6. **Dense:** Capa de salida con 2 neuronas y activación sigmoidal.

El modelo se compila con el optimizador Adam (tasa de aprendizaje de 0.0001) y la función de pérdida `binary_crossentropy`, monitoreando la precisión durante el entrenamiento.

Este modelo CNN está diseñado para detectar manipulaciones en imágenes, capturando características relevantes a través de capas convolucionales y de pooling, y tomando decisiones de clasificación mediante las capas densas.

6.4 Análisis de Resultados

Hemos variado el optimizado y la calidad de recompresión de las imágenes, obteniendo los siguientes resultados:

Optimizador	Tamaño del paso	Calidad de recompresión	Train Acc	Val Acc
Adam	0.001	50	0.98	0.66
Adam	0.001	60	0.99	0.74
Adam	0.001	80	0.97	0.68
RMSprop	0.001	80	0.98	0.73
Adam	0.0001	90	0.997	0.76

Cuadro 2: Resultados de evaluación del modelo con diferentes parámetros

Para todos los entrenamientos se utilizaron 5000 imágenes auténticas y 4500 imágenes editadas, de las cuales, el 20 % se han utilizado como validación y el resto como entrenamiento. Se ha ajustado para tener un máximo de 15 epochs, con una función de callback para detener el entrenamiento en caso de no mejorar el ValAcc en varios epochs.

Como podemos observar se obtuvieron resultados muy cercanos a 1 en el conjunto de entrenamiento, mientras que el modelo tuvo un comportamiento bastante mediocre en el conjunto de validación, lo que indica que está haciendo overfitting sobre el conjunto de entrenamiento.

Para evitar el sobreajuste de parámetros, se probó añadir capas de Dropout al modelo, pero con estas se obtuvieron resultados aún peores.

7 Utilizando Transfer Learning

El *Transfer Learning* (aprendizaje por transferencia) es una técnica en machine learning donde un modelo entrenado en una tarea se reutiliza como punto de partida para una nueva tarea relacionada. Esta técnica es especialmente útil cuando se dispone de un conjunto de datos limitado para la nueva tarea, pero se cuenta con un modelo preentrenado en un conjunto de datos grande y diverso. Las dos formas principales de hacer *transfer learning* son la **extracción de características** y el **fine-tuning** (ajuste fino).

Dado nuestras limitaciones en capacidad de cómputo, hemos optado por utilizar esta técnica buscando maximizar la eficiencia y efectividad de nuestro modelo.

7.1 Extracción de Características

La extracción de características implica utilizar las capas convolucionales de un modelo preentrenado como un extractor de características fijo y entrenar únicamente la(s) capa(s) de clasificación final(es) en el nuevo conjunto de datos. Entre las ventajas de la extracción de características destacan que requiere menos tiempo de entrenamiento ya que solo se entrena unas pocas capas, además es útil cuando el conjunto de datos específico es pequeño.

A continuación, describimos los pasos necesarios para realizar la extracción de características:

1. **Carga del modelo preentrenado:** Se carga un modelo que ha sido preentrenado en un conjunto de datos grande.
2. **Congelación de las capas:** Se congelan todas las capas del modelo, excepto la última capa de clasificación, para que los pesos de estas capas no se actualicen durante el entrenamiento.
3. **Añadir nuevas capas de clasificación:** Se eliminan las capas de clasificación originales del modelo preentrenado y se añaden nuevas capas densas adecuadas para la nueva tarea.
4. **Entrenamiento del nuevo clasificador:** Se entrena solo la nueva parte de clasificación con el conjunto de datos específico de la nueva tarea.

7.2 Ajuste Fino (Fine-Tuning)

El ajuste fino (*fine-tuning*) implica no solo entrenar las nuevas capas de clasificación, sino también ajustar algunos de los pesos de las capas preentrenadas. Generalmente, se descongelan las últimas capas convolucionales del modelo preentrenado y se reentrenan junto con las nuevas capas de clasificación. Esta técnica permite un ajuste más preciso a la nueva tarea, especialmente útil cuando el conjunto de datos específico es moderadamente grande. Además, mejora el rendimiento del modelo en la nueva tarea al permitir que el modelo aprenda características más específicas de los nuevos datos.

A continuación, enumeramos los pasos necesarios para realizar la extracción de características:

1. **Carga del modelo preentrenado:** Se carga un modelo preentrenado en un conjunto de datos grande.
2. **Congelación inicial de las capas:** Inicialmente se congelan todas las capas del modelo para entrenar solo las nuevas capas de clasificación.
3. **Descongelación de algunas capas:** Despues de entrenar las nuevas capas de clasificación, se descongelan las últimas capas convolucionales del modelo preentrenado.
4. **Entrenamiento conjunto:** Se entrena el modelo completo, ajustando tanto las nuevas capas de clasificación como las capas descongeladas del modelo preentrenado.

7.3 Modelo importado

Utilizamos como modelo base el modelo **MobileNetV2**, esta es una arquitectura de red neuronal profunda diseñada principalmente para dispositivos móviles y embebidos con recursos limitados, como potencia de procesamiento y memoria. Esta arquitectura fue presentada en [15] como una mejora de la original MobileNet.

Características Principales

- Bottleneck Residual Blocks: Introduce bloques residuales embotellados, que comprimen y expanden características, reduciendo la dimensionalidad antes de aplicar convoluciones de profundidad separables y luego la expande nuevamente.
- Convoluciones de Profundidad Separable: Divide la convolución estándar en dos operaciones: una convolución de profundidad y una convolución puntual. Reduce el número de parámetros y operaciones computacionales, mejorando la eficiencia.
- ReLU6 Activation: Utiliza la activación ReLU6 en lugar de ReLU, limitada a un rango [0, 6], lo cual es más adecuado para dispositivos móviles debido a sus características de baja precisión y eficiencia.
- Inverted Residuals: Inversión del bloque residual tradicional: en lugar de reducir y luego expandir, primero expande y luego reduce. Facilita el flujo de gradientes y mejora la eficiencia del modelo.

Arquitectura

- Entrada y Primeras Capas: Comienza con una convolución estándar seguida de una convolución de profundidad separable.
- Bloques Residuales Embotellados: Consiste en múltiples bloques con tres convoluciones: expansión (con ReLU6), convolución de profundidad y reducción.
- Últimas Capas: Termina con una capa de convolución de profundidad separable seguida de una capa completamente conectada.

Ventajas

- Eficiencia Computacional: MobileNetV2 tiene un menor número de parámetros y operaciones computacionales comparado con arquitecturas más pesadas, lo que lo hace adecuado para dispositivos móviles con limitaciones de hardware.
- Desempeño Competitivo: A pesar de su eficiencia, MobileNetV2 ofrece un rendimiento competitivo en tareas de visión por computadora, como clasificación de imágenes y detección de objetos.
- Escalabilidad: Esta arquitectura permite ajustar el tamaño del modelo según las necesidades específicas, ofreciendo versiones más pequeñas y eficientes sin sacrificar demasiado rendimiento.

7.4 Resultados obtenidos

Durante el entrenamiento observamos que con 10 epochs de fine tuning y 10 epoch de entrenamiento solo a las capas agregadas, y usando como optimizador RMSprop con un tamaño del paso 0.0001 se obtiene un rendimiento de un 70 % tanto en el conjunto de entrenamiento como en el de validación. Fue observado un lento y constante crecimiento en la precisión en los conjuntos de entrenamiento y prueba, por lo que se decidió probar con más epochs y mayor tamaño del paso, al aumentar el tamaño del paso se obtuvieron peores resultados, al aumentar la cantidad de epochs a 15, la precisión aumenta a 72 %, pero ya se comienzan a ver indicios de overfitting.

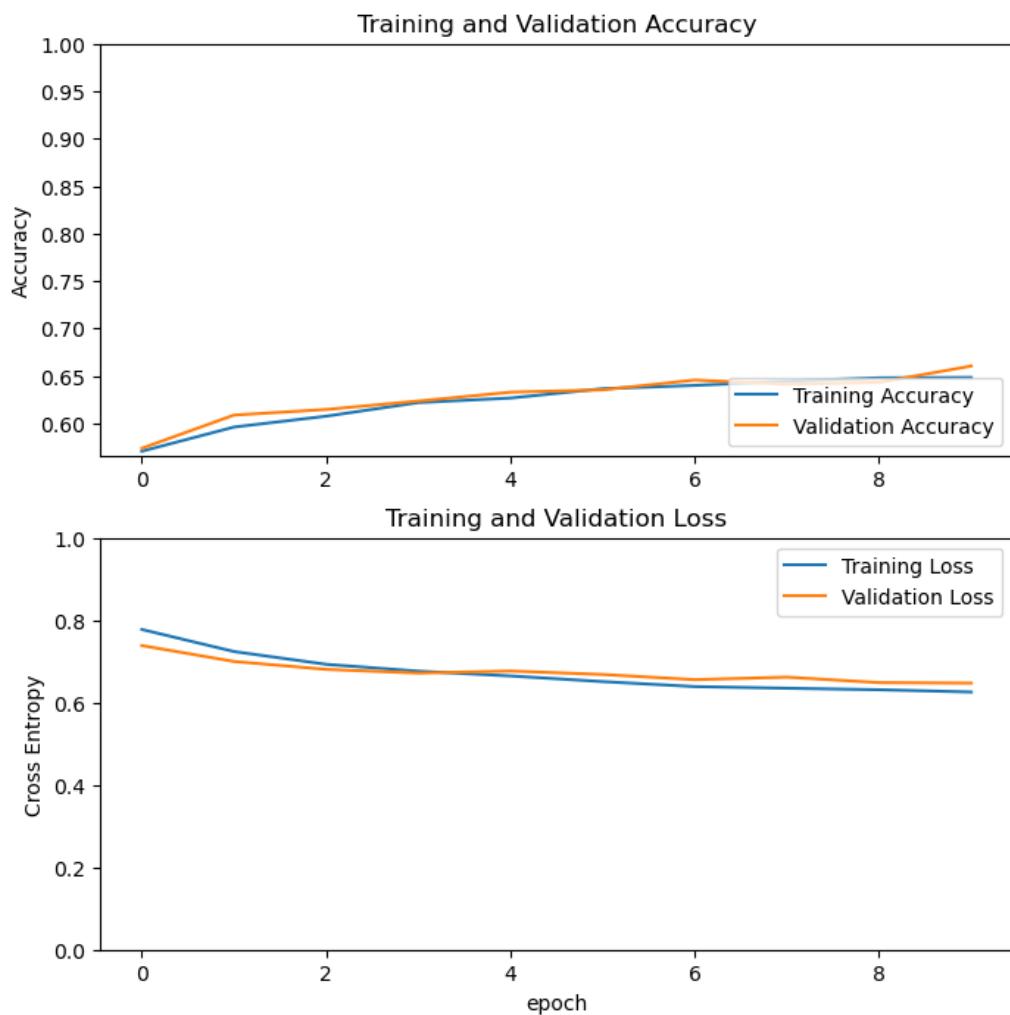


Figura 3

8 Evaluación de algunos métodos no supervisados para el análisis de imágenes

8.1 Utilizando Residual de Interpolación del Canal Verde CFA

El siguiente algoritmo tiene como objetivo detectar áreas sospechosas de manipulación en imágenes mediante el análisis del canal verde de una imagen en formato RGB. Utiliza técnicas basadas en los residuales de interpolación para identificar posibles ediciones. A continuación, se detallan los pasos principales del algoritmo.

8.1.1 Extracción del Canal Verde

Dada una imagen $I \in \mathbb{R}^{m \times n \times 3}$, donde $m \times n$ son las dimensiones de la imagen y los 3 canales corresponden a los colores rojo, verde y azul, se extrae el canal verde:

$$G = I[:, :, 1] \quad (1)$$

Aquí, $G \in \mathbb{R}^{m \times n}$ representa el canal verde de la imagen.

8.1.2 Cálculo del Residual de Interpolación CFA

El siguiente paso es calcular el residual de interpolación utilizando un filtro específico. Primero, se define un kernel de interpolación:

$$K = \begin{bmatrix} 0 & 0,25 & 0 \\ 0,25 & -1 & 0,25 \\ 0 & 0,25 & 0 \end{bmatrix} \quad (2)$$

Este kernel se utiliza para aplicar un filtro a través del canal verde. El residual se define como:

$$R = G * K \quad (3)$$

donde $*$ representa la operación de convolución 2D entre el canal verde G y el kernel K . El resultado $R \in \mathbb{R}^{m \times n}$ representa el residual de interpolación de la imagen.

Al aplicar este kernel, el resultado será cercano a cero si el valor del píxel central está en línea con la interpolación de sus vecinos. En cambio, si ha habido algún tipo de manipulación o inconsistencia, el residual será alto.

8.1.3 Cálculo de la Energía del Residual

La energía del residual se calcula aplicando un filtro Gaussiano sobre el cuadrado del residual:

$$E = \text{GaussianBlur}(R^2, (5, 5), 0) \quad (4)$$

donde $\text{GaussianBlur}(\cdot, (5, 5), 0)$ es una función de suavizado Gaussiano con una ventana de 5×5 y una desviación estándar de 0. La energía del residual $E \in \mathbb{R}^{m \times n}$ resalta regiones con altos valores residuales, lo cual es indicativo de posibles manipulaciones.

8.1.4 Normalización de la Energía del Residual

Para llevar los valores de la energía residual a un rango entre 0 y 1, se normaliza de la siguiente forma:

$$E_{\text{norm}} = \frac{E - E_{\min}}{E_{\max} - E_{\min}} \quad (5)$$

donde E_{\min} y E_{\max} son el valor mínimo y máximo de la energía residual, respectivamente. La energía normalizada $E_{\text{norm}} \in [0, 1]$ facilita la posterior segmentación de las áreas sospechosas.

8.1.5 Detección de Manipulación

Finalmente, se aplica un umbral para detectar las áreas sospechosas. Se define un umbral $T = 0,1$, y se obtiene una máscara binaria que indica las posibles áreas manipuladas:

$$M = \{(i, j) | E_{\text{norm}}(i, j) > T\} \quad (6)$$

donde M es la máscara binaria que indica los píxeles sospechosos de haber sido manipulados. No solo los píxeles con alto residual son indicativo de edición, la formación de áreas oscuras con alto residual alrededor también son indicativo.

8.1.6 Resultados

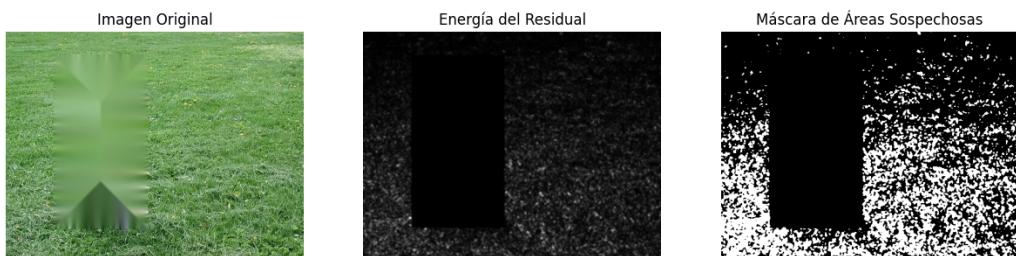


Figura 4

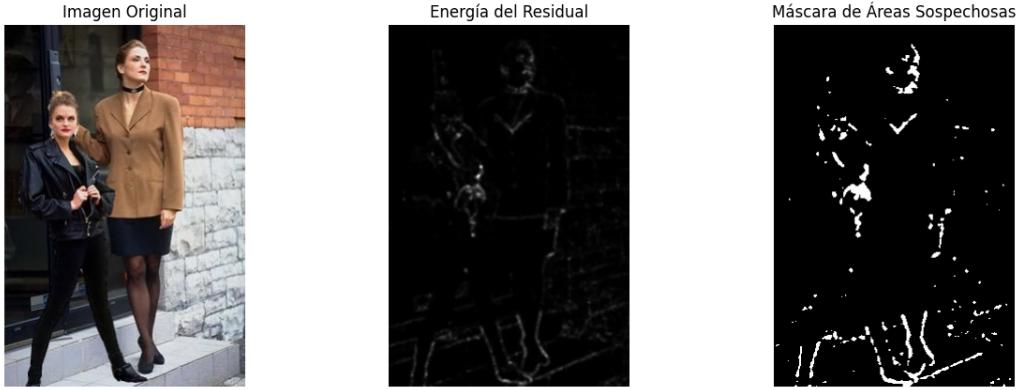
En la figura 4 se puede observar claramente en la imagen original donde ocurrió la edición y podemos notar en la máscara la detección de esta zona.



Figura 5

En la figura 5 se utilizó un modelo mejor para el borrado pero aún así se puede observar donde ocurrió la edición en la máscara.

Los mejores resultados de este modelo son con fotos con zonas que se rellenaron para ocultar la información original, dado que estas zonas tendrán una distribución de residuales anómala que se podrá detectar con el análisis de la máscara.

**Figura 6**

En la figura 6 el tipo de edición es adición (mujer de negro), y vemos como la máscara no nos aporta nada para detectar la edición dado que también resalta áreas auténticas.

8.2 Utilizando el análisis del Ruido PRNU

Este algoritmo tiene como objetivo detectar áreas manipuladas en una imagen analizando el ruido inherente a la cámara, conocido como PRNU (Photo Response Non-Uniformity). El PRNU es un patrón de ruido que es único para cada sensor y puede ser utilizado para determinar si una imagen ha sido modificada.

8.2.1 Descripción del Algoritmo

El algoritmo se divide en tres partes principales: extracción del patrón PRNU, detección de manipulaciones mediante el análisis de la desviación estándar del ruido, y la superposición de una máscara sobre la imagen original para resaltar las áreas sospechosas.

8.2.2 Extracción del Ruido PRNU

El patrón PRNU se extrae mediante el filtrado de alta frecuencia de la imagen:

- La imagen de entrada $I \in \mathbb{R}^{m \times n \times 3}$ se convierte a escala de grises para simplificar el procesamiento:

$$G = \text{cvtColor}(I, \text{GRAYSCALE}) \quad (7)$$

- Para eliminar componentes de baja frecuencia, se aplica un filtro de mediana de 3×3 :

$$G_{\text{denoised}} = \text{medianBlur}(G, 3) \quad (8)$$

- El ruido residual se calcula restando la imagen suavizada de la imagen original:

$$N = G - G_{\text{denoised}} \quad (9)$$

donde $N \in \mathbb{R}^{m \times n}$ representa el patrón de ruido extraído de la imagen.

8.2.3 Detección de Manipulación

La manipulación en la imagen se detecta analizando las inconsistencias en la desviación estándar del ruido N .

- Se divide la imagen en bloques de tamaño $b \times b$, y se define el mapa de desviación estándar $M \in \mathbb{R}^{\frac{m}{b} \times \frac{n}{b}}$.

2. Para cada bloque de la imagen:

$$M_{i,j} = \text{std}(N_{i:i+b,j:j+b}) \quad (10)$$

donde $\text{std}(\cdot)$ representa la desviación estándar del ruido en el bloque $N_{i:i+b,j:j+b}$.

3. Se redimensiona el mapa de desviación estándar para que coincida con el tamaño de la imagen:

$$M_{\text{resized}} = \text{resize}(M, (m, n)) \quad (11)$$

4. Luego, se normaliza el mapa de desviación estándar para obtener valores entre 0 y 1:

$$M_{\text{norm}} = \frac{M_{\text{resized}} - M_{\min}}{M_{\max} - M_{\min}} \quad (12)$$

donde M_{\min} y M_{\max} son el valor mínimo y máximo del mapa, respectivamente.

5. Para detectar las áreas sospechosas, se aplica un umbral al mapa normalizado. Se considera como sospechosa cualquier área con una desviación estándar menor al promedio menos un valor ajustable:

$$T = \text{mean}(M_{\text{norm}}) - 0,05 \quad (13)$$

La máscara resultante es:

$$\text{mask}(i, j) = \begin{cases} 1 & \text{si } M_{\text{norm}}(i, j) < T \\ 0 & \text{en otro caso} \end{cases} \quad (14)$$

8.2.4 Superposición de la Máscara sobre la Imagen Original

Finalmente, se superpone la máscara generada sobre la imagen original para resaltar las áreas sospechosas de manipulación usando un factor de transparencia α .

8.2.5 Resultados

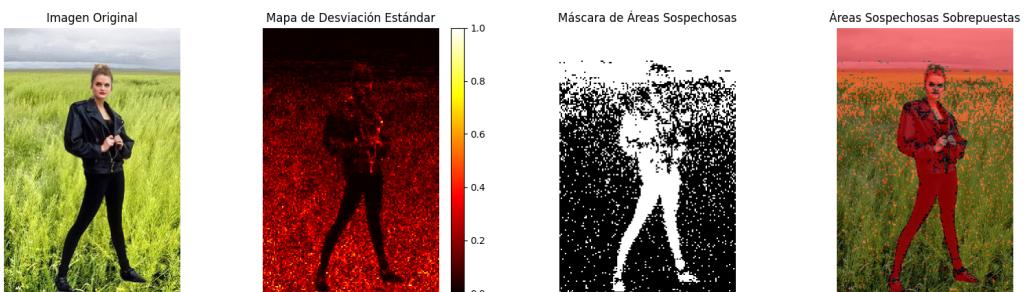
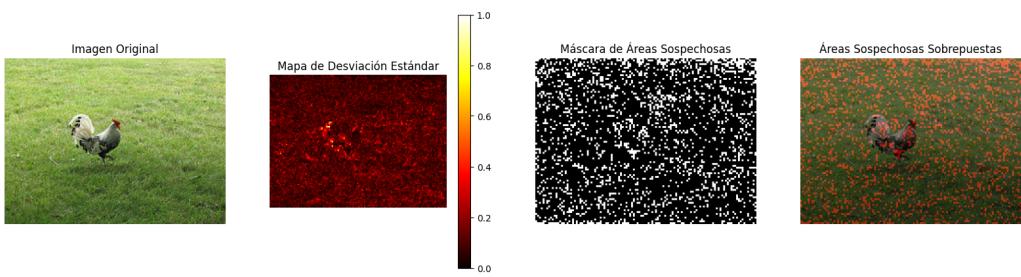
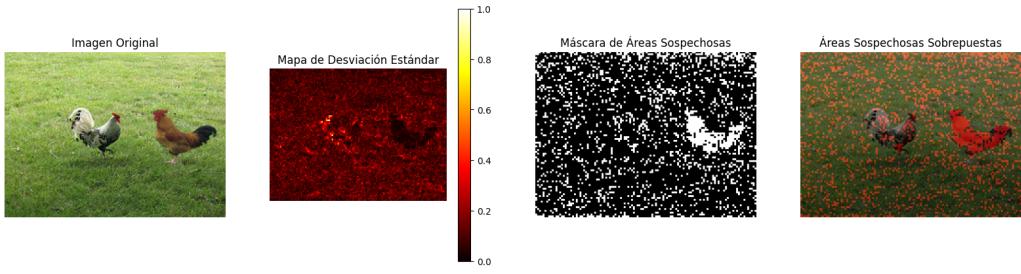


Figura 7

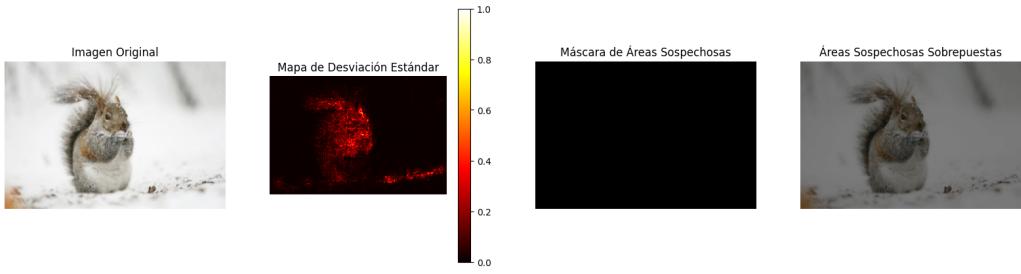
En la figura 7 se nota la adición de la mujer en negro y se aprecia en la máscara superpuesta la correcta segmentación.

**Figura 8**

En la figura 8 la imagen original es un ejemplo similar al anterior, un objeto entre un fondo uniforme, pero la diferencia es que esta imagen es auténtica, nada fue editado, y se puede notar en la máscara como ninguna zona fue segmentada, solo puntos dispersos.

**Figura 9**

En la figura 9 la imagen original es la misma que la anterior pero con un objeto (gallo) añadido, y se puede notar como la máscara segmenta al objeto falso, y al verdadero no.

**Figura 10**

En la figura 10 la imagen original es auténtica y la máscara no segmenta nada en lo absoluto.

Por lo tanto se puede decir tras el análisis de los resultados que este método es especialmente bueno para la detección de adiciones, ya que las zonas añadidas tendrán potencialmente una desviación distinta a la imagen original.

9 Análisis de resultados

A lo largo de este estudio, hemos explorado diversas técnicas para la detección de manipulaciones en imágenes digitales, abordando el problema desde múltiples enfoques. A continuación se muestra un resumen de los resultados obtenidos:

1. Generative Adversarial Network (GAN): Este modelo no logró resultados satisfactorios, ni en la manipulación de imágenes por parte del generador ni en la clasificación de las mismas por parte del discriminador. Este resultado puede deberse a nuestra falta de recursos de cómputo o falta de dataset, de igual manera, es posible que la arquitectura creada no sea la óptima para la tarea en cuestión.
2. U-Net: El modelo implementado mostró una muy lenta velocidad de entrenamiento, por lo que nos planteamos que no disponemos de recursos suficientes para usar el modelo creado o que debemos utilizar técnicas para aumentar la eficiencia del entrenamiento. Por estos motivos, no fue posible entrenarlo con dataset de tamaño significativo y los resultados obtenidos fueron insatisfactorios.
3. Técnicas de Segmentación y Redes Neuronales Convolucionales (CNNs): La idea utilizada para este modelo es basada en el paper [1], aunque en este paper afirman que su modelo posee un rendimiento del 92 % no estuvimos ni remotamente cerca de obtener este resultado, obteniéndose un rendimiento de aproximadamente 67 %. Estos resultados pueden deberse a un mejorable uso de las técnicas de segmentación de imágenes. Además este modelo tiende a hacer overfitting, por lo que sería útil utilizar técnicas de aumento de datos o reunir un mayor dataset.
4. Transfer Learning. El uso de Transfer Learning utilizando extracción de características y Fine Tuning mostró un rendimiento ligeramente inferior a un 70 %. Para obtener mejores resultados deberíamos valorar la posibilidad de utilizar otro modelo preentrenado o entrenar este modelo con una mayor cantidad de datos.
5. Extracción de características y entrenamiento de un modelo SVM: A pesar de utilizar un modelo relativamente antiguo como la Máquina de Soporte Vectorial (SVM), este modelo mostró relativamente buenos resultados, acercándose al 75 %, la mayor de nuestras limitantes para el uso del mismo fue el déficit de memoria RAM, por lo que tuvimos que reducir los datos de entrenamiento.

9.1 Repercusión ética de las soluciones

En la actualidad, donde es tan frecuente la manipulación de información, la detección de edición de imágenes es una tecnología fundamental para garantizar la veracidad y la integridad de la información visual. Estas herramientas juegan un papel crucial en la lucha contra la desinformación y la falsificación digital. Sin embargo, su desarrollo y uso también plantean importantes cuestiones éticas que deben ser consideradas cuidadosamente. En primer lugar, el propósito detrás del uso de herramientas de detección de edición de imágenes debe ser claro y transparente, es esencial que estas tecnologías se utilicen para promover la veracidad y combatir el fraude visual, no para fines malintencionados o invasivos como verificación de imágenes personales en entornos no profesionales. Además, al estas herramientas no ser 100 % precisas, no se debe confiar ciegamente en ellas, ya que esto provocaría consecuencias graves, como acusaciones falsas o la perpetuación de imágenes manipuladas, por lo que los resultados de estas detecciones deben ser interpretados con cautela y en el contexto adecuado, evitando conclusiones precipitadas. Finalmente, la educación y la conciencia sobre las herramientas de detección de edición de imágenes es esencial. Los usuarios y el público en general deben estar informados sobre cómo funcionan estas tecnologías, sus capacidades y limitaciones. La formación adecuada puede ayudar a prevenir malentendidos y fomentar un uso más responsable y ético de estas herramientas.

10 Conclusiones

En el transcurso de nuestra investigación, nos enfrentamos a significativas restricciones de recursos computacionales que impactaron notablemente nuestra capacidad para entrenar modelos de alta complejidad. Arquitecturas avanzadas como las Redes Generativas Adversarias (GAN) y las redes U-Net, conocidas por su potencial en tareas de procesamiento de imágenes, no pudieron ser explotadas en su totalidad debido a estas limitaciones. Como resultado, el rendimiento general de estos modelos quedó por debajo de las expectativas iniciales.

No obstante, es importante destacar que, a pesar de estas restricciones, algunos modelos demostraron un desempeño prometedor. En particular, el modelo de Máquinas de Vectores de Soporte (SVM) sobresalió, alcanzando una precisión cercana al 75 %. Este resultado es prometedor dadas las circunstancias y sugiere que, con los recursos adecuados, modelos más complejos podrían alcanzar resultados aún más alentadores.

En conclusión, aunque nos enfrentamos a desafíos significativos, los resultados obtenidos sientan las bases para futuras investigaciones. Con las mejoras propuestas, esperamos que futuros trabajos en este campo puedan contribuir significativamente a la lucha contra la desinformación visual, manteniendo al mismo tiempo un equilibrio entre la innovación tecnológica y la responsabilidad social.

11 Propuestas para futuros trabajos

Basándonos en los resultados y limitaciones de nuestro estudio actual, proponemos las siguientes líneas de investigación y mejoras para futuros trabajos:

1. **Ampliación y diversificación del dataset:** Incrementar el tamaño y variedad del dataset, incluyendo una mayor diversidad de tipos de falsificaciones. Implementar técnicas avanzadas de aumento de datos, como GANs, para generar ejemplos sintéticos realistas y mejorar la generalización del modelo.
2. **Mejora en la extracción de características:** Implementar técnicas de segmentación de imágenes basadas en aprendizaje profundo y explorar arquitecturas CNN más avanzadas, como ResNet, para una extracción más precisa y eficiente de características relevantes.
3. **Optimización de recursos computacionales:** Es crucial explorar estrategias para optimizar el uso de recursos computacionales.
4. **Mejora de la interpretabilidad:** Desarrollar métodos para visualizar y explicar las áreas de la imagen que el modelo considera sospechosas, aumentando la confiabilidad y aplicabilidad del sistema.
5. **Evaluación en escenarios del mundo real:** Probar el modelo en imágenes recolectadas de redes sociales y medios de comunicación, desarrollando métricas de evaluación que reflejen mejor el rendimiento en situaciones prácticas.

Referencias

- [1] Schemm, T., Srivastava, A., Singh, A., & Rathod, M. (2022). A Novel Approach to Predictive Maintenance Using Machine Learning. *Electronics*, 11(3), 403.
- [2] Sharma, N. B. (2021, February 1). TensorFlow: Classify Images of Cats and Dogs by Using Transfer Learning. *Medium*.
- [3] Huh M, Liu A, Owens A, Efros AA (2018) Fighting fake news: image splice detection via learned self-consistency.
- [4] Zhang, Y., Goh, J., Win, L. L., Thing, V. (2016). Image Region Forgery Detection: A Deep Learning Approach. In Proceedings of the Singapore Cyber-Security Conference (SG-CRC) 2016 (pp. 1-11). *Cryptology and Information Security Series, Vol. 14*
- [5] Reddy V, Vaghdevi K, Kolli D. (2021). Digital image forgery detection using SUPER pixel segmentation and HYBRID feature point mapping. *European J Molecular Clin Med*
- [6] ISO/IEC 10918-1:1992 – Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines
- [7] Segmentacion de imagenes deramtoscopicas usando la arquitectura UNET. Tesis de Grado Marcos Adrian Valdivie Rodriguez. Facultad de Matematica y Computacion. Universidad de La Habana 2022
- [8] B. Liu, R. Wu, X. Bi, B. Xiao, W. L, G. Wang, and X. Gao. D-Unet A dual encoder U-Net for image splicing forgery detection and localization, 2020
- [9] Z. Zhou. Unet Redesigning Skip Conections to exploit multiscale features in image segmentation
- [10] Y. Rao and J. Ni, “A deep learning approach to detection of splicing and copy-move forgeries in images,” in Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS), Dec. 2016, pp. 1–6.
- [11] B. Xiao, Y. Wei, X. Bi, W. Li, and J. Ma, “Image splicing forgery detection combining coarse to refined convolutional neural network and adaptive clustering,” Inf. Sci., vol. 511, pp. 172–191, Feb. 2020
- [12] L. Bondi, S. Lameri, D. Guera, P. Bestagini, E. J. Delp, and S. Tubaro, Tampering detection and localization through clustering of camera based CNN features, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jul. 2017, pp. 1855–1864.
- [13] L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, First steps toward camera model identification with convolutional neural networks, IEEE Signal Process. Lett., vol. 24, no. 3, pp. 259–263, Mar. 2017.
- [14] Image Forgery Detection using deep learning by recompressiong images, I.I. Ganapathi, N.S. Vu, S.D. Ali, N.Werghi, N. Saxena
- [15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 4510–4520.
- [16] Wu, Y., AbdAlmageed, W., Natarajan, P. (2019). ManTra-Net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9543-9552). [enlace](#)
- [17] Fischinger, D., Boyer, M. (2023). DF-Net: The Digital Forensics Network for Image Forgery Detection. En Irish Machine Vision and Image Processing Conference. Zenodo. [enlace](#)
- [18] Wu, H., Chen, Y., Zhou, J. (2023). Rethinking Image Forgery Detection via Contrastive Learning and Unsupervised Clustering. arXiv. [enlace](#)