



MACHINE LEARNING

PROYECTO FINAL

Detección de edición en imágenes

Autores:

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

danielabad774@gmail.com
enzord2001@gmail.com

2024

1 Introducción

En el presente proyecto, 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 proyecto 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, entre otros.

El análisis comparativo de estos métodos nos permitirá identificar las estrategias más efectivas para la detección de ediciones en imágenes y comprender las fortalezas y debilidades de cada enfoque en diferentes escenarios y condiciones. Al final del proyecto, esperamos proporcionar una solución integral y bien fundamentada que pueda ser aplicada en diversos contextos, contribuyendo así al avance de las técnicas de detección de manipulación de imágenes en el ámbito digital.

Este proyecto representa no solo un desafío técnico significativo, sino también una oportunidad para aplicar los conocimientos adquiridos a lo largo de nuestra formación en 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.

Índice

1. Introducción	1
2. Estado del Arte	4
2.1. Métodos Basados en Características	4
2.1.1. Análisis de Metadatos	4
2.1.2. Análisis de Inconsistencias Visuales	4
2.2. Métodos Basados en Machine Learning	4
2.2.1. Redes Neuronales Convolucionales (CNN)	4
2.2.2. Autoencoders	4
2.2.3. Redes Generativas Antagónicas (GAN)	4
2.3. Técnicas Avanzadas	4
2.3.1. Transfer Learning	4
2.3.2. Análisis Forense de Imágenes	4
2.3.3. Detección Basada en Huellas Dactilares de Imagen	5
2.4. Desafíos y Tendencias Futuras	5
2.4.1. Desafíos	5
2.4.2. Tendencias Futuras	5
3. Dataset utilizado	6
3.1. Características del CASIA V2	6
3.2. Uso en la Investigación	6
3.3. Aplicaciones	6
4. Extracción de Características y Entrenamiento de un Modelo SVM	8
4.1. Carga de Imágenes	8
4.2. Extracción de Características	8
4.2.1. Transformada de Fourier	8
4.2.2. Características de Ruido	8
4.2.3. Detección de Bordes	8
4.2.4. Características de Textura	8
4.2.5. Segmentación de Imágenes	8
4.3. Preparación del Conjunto de Datos	8
4.4. Entrenamiento del Modelo SVM	8
4.5. Resultados	9
4.5.1. Selección del Conjunto de Datos y Enfoque del Análisis	9
4.5.2. Precisión General del Modelo	9
4.5.3. Análisis de la Importancia de las Características	9
4.5.4. Pruebas Adicionales	9
4.5.5. Conclusiones del Análisis de Características	9
5. Atacando el problema utilizando técnicas básicas de Deep Learning	10
5.1. Utilizando U-Nets	10
5.1.1. Etapa de Contracción	10
5.1.2. Etapa de Expansión	10
5.1.3. Skip Connections	10
5.1.4. Resultados	11
5.2. Utilizando GAN	12
5.2.1. Metodología	12
5.2.2. Configuración del Generador	12
5.2.3. Configuración del Discriminador	12
5.2.4. Resultados	13
5.2.5. Conclusiones	13

6. Utilizando técnicas de segmentación de imágenes y una CNN	14
6.1. Enfoque seguido	14
6.2. Modelo utilizado	14
6.2.1. Capas del Modelo	14
6.2.2. Compilación del Modelo	15
6.2.3. Resumen del Modelo	15
6.3. Análisis de Resultados	15
7. Atacando el problema utilizando técnicas avanzadas de Deep Learning	17
7.1. Transfer Learning	17
7.1.1. Extracción de Características	17
7.1.2. Fine Tuning	17
7.1.3. Modelo importado	18
7.2. Resultados obtenidos	19
8. Combinando modelos	20
8.1. ¿Qué es Boosting?	20
8.2. ¿Cómo Funciona el Boosting?	20
8.3. Ventajas del boosting	20
8.4. Implementación	20
8.4.1. Modelos Base Seleccionados	20
8.5. Resultados obtenidos	21
9. Análisis de resultados	22
9.1. Repercusión ética de las soluciones	22
10. Conclusiones	23
11. Propuestas para futuros trabajos	24
12. Referencias	25

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 Desafíos y Tendencias Futuras

2.4.1 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.

2.4.2 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.

Cuadro 1: Revisión Bibliográfica (Basado en Compresión)

Área de Detección de Falsificaciones	Investigadores	Año	Enfoque	Dataset	Desempeño
Detect tampered images in different image formats	Zhang et al. [4]	2016	Modelo de autoencoder apilado (SAE) (3 capas) Característica de entrada: DCT	CASIA	Precisión: 91.09 % (En general para JPEG y TIFF) Tasa de falsos positivos: 4.31 %
Image Forgery Detection Using Deep Learning by Recompressing Images	Ali et al. [1]	2022	Técnica de recompresión de imágenes con CNN	CASIA 2.0	Precisión: 92.23 % (98 % calidad), F1-score: 91.08 %

Cuadro 2: Revisión Bibliográfica (Imágenes Basadas en Píxeles)

Área de Detección de Falsificaciones	Investigadores	Año	Enfoque	Dataset	Desempeño
Image splice detection via learned self-consistency.	Huh et al [3]	2018	CNN Input Feature: EXIF metadata, pixel values Input Size: 128×128 Background Architecture: ResNet-v2	ad-hoc, Carvalho, Columbia Gray, Realistic (Korus), On-the-wild websites.	
Super Pixel Segmentation and Hybrid Feature Point Mapping for Digital Image Forgery Detection	Reddy et al [5]	2021	ResNet (CNN Based), SIFT, SURF Input Size: 128×128 Background Architecture: Own	ad-hoc	Más robusto que los métodos convencionales

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:

4.2.1 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.

4.2.2 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.

4.2.3 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.

4.2.4 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.

4.2.5 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

4.5.1 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.

4.5.2 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.

4.5.3 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 3: Impacto de la eliminación de características individuales en la precisión del modelo SVM.

4.5.4 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.

4.5.5 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 Atacando el problema 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.

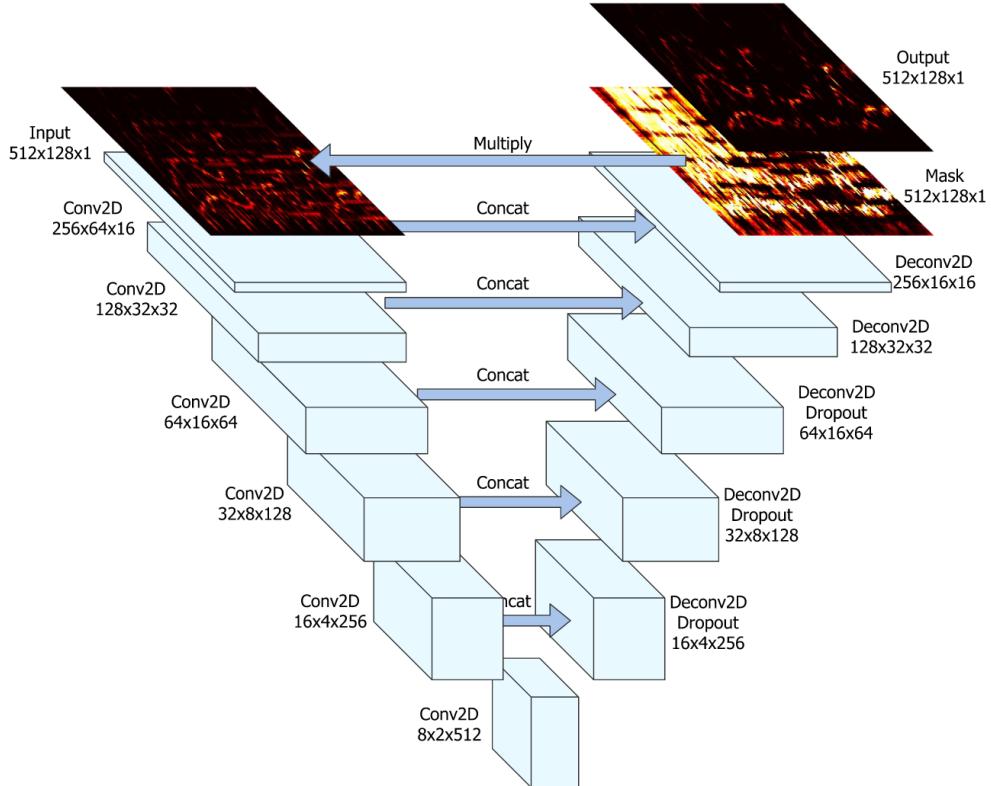


Figura 1: Arquitectura de una U-Net típica.

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 GAN

5.2.1 Metodología

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.

5.2.2 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.

5.2.3 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.

5.2.4 Resultados

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

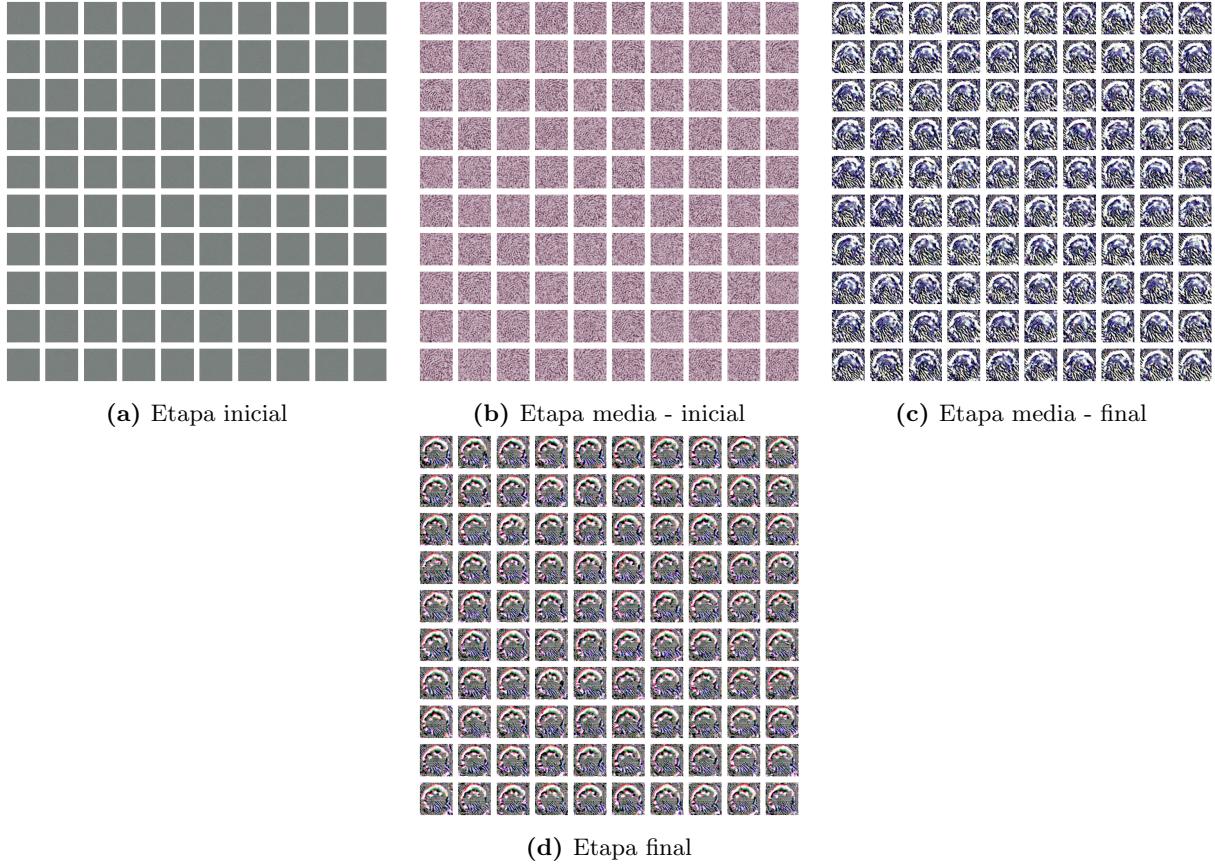


Figura 2: Evolución de las imágenes generadas por el generador en diferentes etapas del entrenamiento.

5.2.5 Conclusiones

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 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 ($A_{recompressed}$)
 3. Calcula la diferencia entre A y $A_{recompressed}$ (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.

6.2 Modelo utilizado

El modelo `Image_Forgery_Predictor_Model` es una red neuronal convolucional (CNN) diseñada para detectar manipulación en imágenes. La arquitectura del modelo se define utilizando la biblioteca Keras con TensorFlow como backend.

6.2.1 Capas del Modelo

1. **Input:** Define la forma de entrada de las imágenes. En este caso, las imágenes son en escala de grises (1 canal) con una resolución de 128x128 píxeles.

```
Input(shape=(128, 128, 1))
```

2. **Conv2D:** Tres capas convolucionales secuenciales con 32 filtros y un tamaño de kernel de 3x3, utilizando la función de activación ReLU.

```
Conv2D(32, (3, 3), activation='relu')
Conv2D(32, (3, 3), activation='relu')
Conv2D(32, (3, 3), activation='relu')
```

3. **MaxPooling2D:** Una capa de max-pooling con un tamaño de pool de 2x2, que reduce la dimensionalidad de las características aprendidas.

```
MaxPooling2D(pool_size=(2, 2))
```

4. **Flatten**: Esta capa aplana las características extraídas por las capas convolucionales en un vector unidimensional.

```
Flatten()
```

5. **Dense**: Una capa densa totalmente conectada con 256 neuronas y función de activación ReLU.

```
Dense(256, activation='relu')
```

6. **Dense**: La capa de salida con 2 neuronas y función de activación sigmoidal, que produce las probabilidades de las dos clases (manipulada o no manipulada).

```
Dense(2, activation='sigmoid')
```

6.2.2 Compilación del Modelo

El modelo se compila utilizando el optimizador Adam con una tasa de aprendizaje de 0.0001. La función de pérdida utilizada es `binary_crossentropy`, adecuada para problemas de clasificación binaria, y se monitorea la métrica de precisión durante el entrenamiento. Fueron probados distintos optimizadores y tamaños de pasos.

6.2.3 Resumen del Modelo

A continuación, se presenta el código completo del modelo:

```
def Image_Forgery_Predictor_Model():
    model = Sequential([
        Input(shape=(128, 128, 1)),
        Conv2D(32, (3, 3), activation='relu'),
        Conv2D(32, (3, 3), activation='relu'),
        Conv2D(32, (3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),
        Flatten(),
        Dense(256, activation='relu'),
        Dense(2, activation='sigmoid')
    ])
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

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 capas densas.

6.3 Análisis de Resultados

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

Nota: Para todos los entrenamientos se han utilizado 5000 imágenes auténticas y 4500 imágenes editadas, de las cuales, el 20 % se han utilizado como validación en 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.

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 4: Resultados de evaluación del modelo con diferentes parámetros

Como podemos observar se obtuvieron resultados muy cercanos a 1 en el conjunto de entrenamiento, mientras que el modelo tuvo un comportamiento 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 Atacando el problema utilizando técnicas avanzadas de Deep Learning

7.1 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).

7.1.1 Extracción de Características

Definición: 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.

Pasos:

1. **Carga del modelo preentrenado:** Se carga un modelo que ha sido preentrenado en un conjunto de datos grande, como ImageNet.
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.

Ventajas:

- Requiere menos tiempo de entrenamiento ya que solo se entranan unas pocas capas.
- Es útil cuando el conjunto de datos específico es pequeño.

7.1.2 Fine Tuning

Definición: El *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.

Pasos:

1. **Carga del modelo preentrenado:** Se carga un modelo preentrenado en un conjunto de datos grande, como ImageNet.
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.

Ventajas:

- Permite un ajuste más preciso a la nueva tarea, especialmente útil cuando el conjunto de datos específico es moderadamente grande.

- Mejora el rendimiento del modelo en la nueva tarea al permitir que el modelo aprenda características más específicas de los nuevos datos.

7.1.3 Modelo importado

Se ha utilizado 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 por Google en 2018 como una mejora de la original MobileNet.

Características Principales

- Bottleneck Residual Blocks
 - Introduce bloques residuales embotellados, que comprimen y expanden características.
 - Reduce 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:
 - Menor número de parámetros y operaciones computacionales comparado con arquitecturas más pesadas.
 - Adecuado para dispositivos móviles con limitaciones de hardware.
- Desempeño Competitivo:
 - A pesar de su eficiencia, ofrece un rendimiento competitivo en tareas de visión por computadora, como clasificación de imágenes y detección de objetos.
- Escalabilidad:
 - Permite ajustar el tamaño del modelo según las necesidades específicas, ofreciendo versiones más pequeñas y eficientes sin sacrificar demasiado rendimiento.

7.2 Resultados obtenidos

Durante el entrenamiento se observó que con 10 epoch 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.

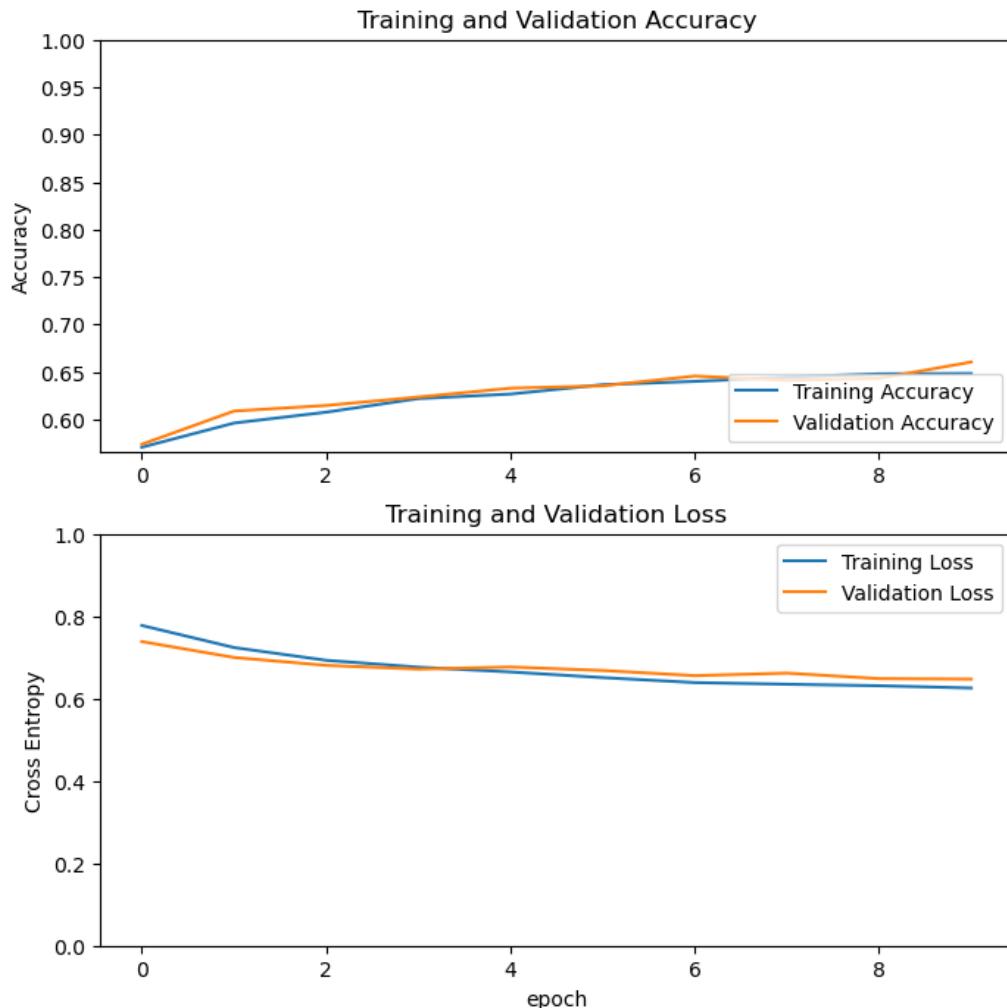


Figura 3

8 Combinando modelos

Finalmente decidimos crear un ensemble entre los modelos creados con el objetivo de mejorar los resultados obtenidos. En particular decidimos usar boosting.

8.1 ¿Qué es Boosting?

El boosting es una técnica de ensemble que crea un modelo fuerte a partir de una serie de modelos débiles, donde cada modelo débil es un clasificador que apenas supera la precisión de un clasificador aleatorio. A diferencia de otros métodos de ensemble como el bagging, donde los modelos se entran de manera independiente, en el boosting los modelos se entran secuencialmente.

8.2 ¿Cómo Funciona el Boosting?

1. Inicialización: Se comienza entrenando un modelo débil (por ejemplo, un árbol de decisión pequeño) utilizando el conjunto de datos original.
2. Pesado de Errores: Se evalúa el modelo y se aumentan los pesos de las muestras mal clasificadas. Esto hace que el siguiente modelo ponga más atención en estas muestras difíciles.
3. Entrenamiento Secuencial: Se entrena un nuevo modelo débil utilizando los datos ponderados, y este proceso se repite. Cada nuevo modelo intenta corregir los errores de los modelos anteriores.
4. Combinación de Modelos: Los modelos entrenados se combinan mediante un método de votación ponderada o una combinación aditiva, donde los modelos que tienen mejor rendimiento tienen un peso mayor en la decisión final.

8.3 Ventajas del boosting

1. Mejora en la Precisión: Al corregir secuencialmente los errores, los modelos de boosting tienden a tener una precisión superior en comparación con los modelos individuales.
2. Flexibilidad: Puede utilizarse con diferentes tipos de modelos base (árboles de decisión, redes neuronales, etc.).
3. Robustez a overfitting: Aunque el boosting puede ser propenso a sobreajustar, técnicas como el regularization y el uso adecuado de hiperparámetros ayudan a mitigar este problema.

8.4 Implementación

Hemos desarrollado un algoritmo de boosting inspirado en AdaBoost para combinar las fortalezas de tres de nuestros modelos más "prometedores". Esta técnica nos permite aprovechar las capacidades complementarias de diferentes enfoques de detección de falsificaciones.

8.4.1 Modelos Base Seleccionados

Los modelos base elegidos para nuestro ensemble son:

- Modelo SVM con extracción de características (Capítulo 4): Este modelo, con una precisión aproximada del 75 %, utiliza técnicas de extracción de características específicas para la detección de falsificaciones, combinadas con una máquina de vectores de soporte (SVM) para la clasificación.
- Modelo CNN con recompresión (Capítulo 6): Este enfoque, que alcanza un rendimiento ligeramente inferior al 70 %, emplea una red neuronal convolucional (CNN).
- Modelo de Transfer Learning (Capítulo 7): También con un rendimiento cercano al 70 %, este modelo aprovecha el conocimiento previo de redes preentrenadas, adaptándolas a nuestra tarea específica de detección de falsificaciones.

Es importante notar que, aunque desarrollamos modelos GAN y U-Net como parte de nuestra investigación, estos no se incluyeron en el ensemble final debido a su largo tiempo de entrenamiento y resultados subóptimos en comparación con los otros modelos.

8.5 Resultados obtenidos

Durante el entrenamiento de este modelo nos enfrentamos a significativas limitaciones computacionales que restringieron nuestra capacidad para entrenarlo con el conjunto de datos completo. Estas restricciones nos obligaron a utilizar solo una pequeña fracción del dataset disponible, lo cual tuvo un impacto directo en la calidad de los resultados obtenidos generalmente bastante mediocres.

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. En total fueron entrenados 6 modelos distintos:

1. Generative Adversarial Network: 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 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 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.
6. Utilización de los tres últimos modelos mencionados para la creación de un ensemble: El uso de boosting para la creación de un modelo fuerte a partir de estos modelos más débiles no ha podido ser sujeto a experimentación con volúmenes de datos significativos.

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 son esenciales. 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 alentador, considerando las circunstancias, y sugiere que con recursos adecuados, los modelos más complejos como el boosting podrían lograr resultados aún más impresionantes.

En conclusión, aunque nos enfrentamos a desafíos significativos, los resultados obtenidos son prometedores y 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.

12 Referencias

Referencias

- [1] Schemm, T., Srivastava, A., Singh, A., & Rathod, M. (2022). A Novel Approach to Predictive Maintenance Using Machine Learning. *Electronics*, 11(3), 403. Recuperado de <https://www.mdpi.com/2079-9292/11/3/403>
- [2] Sharma, N. B. (2021, February 1). TensorFlow: Classify Images of Cats and Dogs by Using Transfer Learning. *Medium*. Recuperado de <https://medium.com/@nutanbhogendrasharma/tensorflow-classify-images-of-cats-and-dogs-by-using-transfer-learning-59da26723bda#:~:text=TensorFlow%20Classify%20Images%20of%20Cats%20and%20Dogs%20by%20Freeze%20the%20convolutional%20base%20...%20M%C3%A1s%20elementos>
- [3] Huh M, Liu A, Owens A, Efros AA (2018) Fighting fake news: image splice detection via learned self-consistency. Recuperado de <https://arxiv.org/pdf/1805.04096.pdf>
- [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* Recuperado de <https://ebooks-iospress.nl/publication/42049>
- [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* https://scholar.google.com/scholar_lookup?journal=European+J+Molecular+Clin+Med&title=Digital+image+forgery+detection+using+SUPER+pixel+segmentation+and+HYBRID+feature+point+mapping&author=V+Reddy&author=K+Vaghdevi&author=D+Kolli&volume=8&issue=2&publication_year=2021&pages=1485-1500&