

Detección de imágenes o vídeos modificados mediante Redes Neuronales

Miguel del Arco Marquez

Resum– Este artículo se centra en la utilización de Deep Learning para la detección de modificaciones en imágenes. Se propone un modelo basado en redes neuronales convolucionales y se utiliza un dataset que ha sido cuidadosamente tratado para garantizar un equilibrio adecuado de muestras por clase. El modelo es entrenado para distinguir entre imágenes originales y modificadas, y muestra resultados precisos, lo que lo convierte en una herramienta valiosa para el análisis forense. El enfoque de Deep Learning permite capturar patrones sutiles y complejos en las imágenes, mejorando la capacidad de detección. Además, el modelo demuestra ser robusto frente a diversas manipulaciones y tiene aplicaciones relevantes en el campo de la forense digital y la verificación de autenticidad. Se sugieren investigaciones futuras para mejorar el modelo y abordar desafíos adicionales en la detección de modificaciones en imágenes.

Palabras claves– Aprendizaje profundo, Detección de imágenes modificadas, Conjuntos de datos, Preprocesamiento de imágenes, Detección de manipulación específica.

Abstract– This article focuses on the application of Deep Learning techniques for the detection of image modifications. A model based on convolutional neural networks is proposed, utilizing a carefully curated dataset with balanced samples across classes. The model is trained to distinguish between original and modified images, yielding accurate results and providing a valuable tool for forensic analysis. Deep Learning enables the capture of subtle and complex patterns in images, enhancing the detection capability. The model demonstrates robustness against various manipulations and finds applications in digital forensics and authenticity verification. Future research directions are suggested to improve the model and address additional challenges in image modification detection.

Keywords– Deep learning, Detection of modified images, Data sets, Image preprocessing, Detection of specific manipulation.

1 INTRODUCCIÓN - CONTEXTO DEL TRABAJO

EN la actualidad, el uso de imágenes en la comunicación y en la toma de decisiones es cada vez más común. Sin embargo, existe la posibilidad de que algunas de estas imágenes sean manipuladas o modificadas para engañar o tergiversar la información que se presenta. Por esta razón, la detección de imágenes modificadas se ha convertido en una tarea importante en el campo de la inteligencia artificial.

Para abordar esta problemática, se han desarrollado di-

versas técnicas basadas en el aprendizaje profundo que permiten detectar si una imagen ha sido modificada o no. El aprendizaje profundo, también conocido como deep learning, es una rama de la inteligencia artificial que utiliza redes neuronales artificiales para aprender y realizar tareas complejas, como el reconocimiento de objetos y el procesamiento de lenguaje natural.

En este trabajo, se abordará la detección de imágenes modificadas mediante el uso de técnicas de aprendizaje profundo. Se discutirán los diferentes modelos de deep learning que se pueden utilizar para este fin, así como los pasos necesarios para llevar a cabo el entrenamiento y la evaluación de los modelos. Asimismo, se destacará la importancia de contar con un conjunto de datos etiquetado y representativo para el entrenamiento de los modelos, así como la necesidad de ajustar y actualizar periódicamente el modelo para mejorar su rendimiento.

En definitiva, este trabajo tiene como objetivo brindar una visión general de cómo se puede utilizar el aprendizaje pro-

- E-mail de contacto: migueldemollet10@gmail.com
- Mención realizada: Computación.
- Trabajo tutorizado por: Jordi Serra Ruiz (Departamento de Ciencias de la Computación)
- Curs 2022/23

fundo para detectar imágenes modificadas y destacar la relevancia de esta tarea en la actualidad.

2 OBJETIVOS

Los objetivos que supone un trabajo de esta magnitud son los siguientes.

1. Identificar y evaluar los modelos de aprendizaje profundo más efectivos para detectar imágenes modificadas en un conjunto de datos específico.
2. Implementar y entrenar un modelo de aprendizaje profundo para la detección de imágenes modificadas, utilizando un conjunto de datos representativo.
3. Analizar la efectividad del modelo en la detección de diferentes tipos de modificaciones, como la manipulación de la información visual, la eliminación de objetos o la inserción de objetos.
4. Comparar el rendimiento del modelo de aprendizaje profundo con otras técnicas de detección de imágenes modificadas, como el análisis forense de imágenes o la detección de patrones.
5. Proporcionar recomendaciones para mejorar la precisión y la eficacia del modelo de aprendizaje profundo en la detección de imágenes modificadas, como la incorporación de nuevos datos de entrenamiento o la adaptación del modelo a nuevos tipos de modificaciones.

3 METODOLOGÍA

Durante el desarrollo del proyecto, se utilizará GitHub como herramienta de gestión de versiones y almacenamiento del código. Esto permitirá tener un seguimiento detallado del progreso del trabajo y una trazabilidad adecuada de los cambios realizados. Además, se ha establecido un proceso de revisión regular con una frecuencia de 15 días, en el cual se compartirá el avance del trabajo con el tutor designado, Jordi Serra Ruiz, del departamento de Ciencias de la Computación, y se recibirá feedback para asegurar que se está avanzando adecuadamente hacia los objetivos del TFG. Este enfoque garantizará una comunicación fluida entre el estudiante y el tutor, lo que resultará en un trabajo de alta calidad.

4 ESTADO DEL ARTE

El campo de la detección de imágenes modificadas mediante el uso de técnicas de aprendizaje profundo se encuentra en constante evolución y desarrollo. A continuación, se presentan algunos de los avances más relevantes y actuales en esta área:

- **Herramientas de pago:** Existe una herramienta que podemos encontrar en la suite de *Adobe Analytics* [1, 2], esta pretende detectar si una imagen ha sido modificada mediante algoritmos de Deep Learning.

Esta tecnología permite a los creadores de contenido proteger sus derechos de autor y verificar la autenticidad de la imagen. Por otro lado permite detectar si dicha imagen es original o ha sido modificada.

Para tener una mejor idea de cómo esta funciona, podemos observar cómo en la figura 1 esta es capaz de detectar objetos añadidos a imágenes.

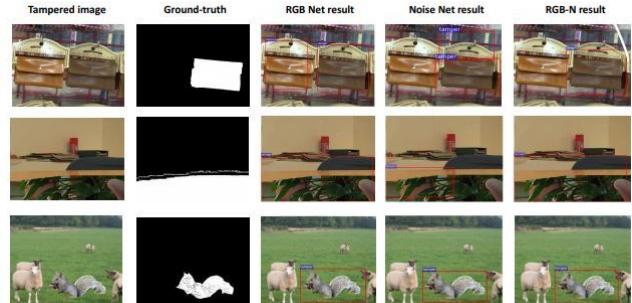


Fig. 1: Ejemplo de funcionamiento de la IA de Adobe

- **Redes Neuronales Convolucionales (CNN):** Las redes neuronales convolucionales son el modelo de deep learning más utilizado en la detección de imágenes modificadas, y han demostrado tener un alto nivel de precisión en la detección de imágenes modificadas por scripts de *Adobe Photoshop*®.

Podemos ver trabajos ya realizados en el cual usan este tipo de red neuronal, un caso sería el trabajo de **Detecting Photoshopped Faces by Scripting Photoshop** [3].



Fig. 2: Resultados Detecting Photoshopped Faces by Scripting Photoshop

Para entender mejor la figura 2 debemos de ver que esta imagen está compuesta compuesta por 3 sub-imágenes de izquierda a derecha sería lo siguiente.

1. Imagen original
2. Imagen modificada
3. Output del modelo

- **Uso de tecnologías existentes como Deep Fake:** Es una técnica de inteligencia artificial que se utiliza para crear videos o imágenes manipuladas que parecen ser auténticas, pero en realidad son falsas. Esta técnica utiliza algoritmos de aprendizaje profundo para entrenar modelos de redes neuronales que pueden analizar, sintetizar y manipular el contenido de una imagen o un video. Con esta técnica, es posible crear videos y fotos que parecen ser reales, pero que en realidad son

el resultado de la manipulación de contenido existente, como el rostro de una persona. Los deepfakes se han utilizado en algunos casos para crear noticias falsas, difamar a personas o para fines de entretenimiento. Es importante destacar que los deepfakes pueden ser utilizados de manera engañosa, por lo que es importante ser cautelosos al consumir contenido generado por esta técnica [4].

Para entender mejor esta técnica podemos ver en la figura 3 el resultado de aplicar esta técnica.



Fig. 3: Resultado del Deep Fake (Output—Input)

Existen papers como **FaceForensics++ - Learning to Detect Manipulated Facial Images** [5] en el cual trata sobre la preocupación creciente acerca de la capacidad de generar y manipular imágenes sintéticas con un alto grado de realismo, lo que puede tener implicaciones graves en la sociedad, ya que puede conducir a una pérdida de confianza en el contenido digital y difundir información falsa o noticias falsas.

Este propone la detección de manipulaciones faciales y da un benchmark automatizado para la evaluación de la eficacia de los métodos de detección.

El benchmark se basa en representantes destacados de manipulaciones faciales y contiene una base de datos de más de 1.8 millones de imágenes manipuladas. Los autores muestran que el uso de conocimiento específico del dominio mejora significativamente la detección de falsificaciones y supera claramente a los observadores humanos. En resumen, el trabajo aborda un problema importante en el ámbito de la generación y manipulación de imágenes sintéticas y propone una solución útil para la detección de manipulaciones faciales.

En la figura 4 podemos observar el Pipeline del proyecto.



Fig. 4: Pipeline FaceForensics++

El estado del arte en la detección de imágenes modificadas mediante el uso de técnicas de aprendizaje profundo se centra en el desarrollo y mejora de los modelos existentes, la exploración de nuevas técnicas de aprendizaje profundo, el uso de grandes conjuntos de datos de entrenamiento y la investigación en detección de manipulación específica.

5 DATASET

La elección de un conjunto de datos adecuado es una parte crítica en la creación de un modelo. La calidad y la representatividad de los datos de entrenamiento son factores clave en la precisión y la eficacia del modelo.

Este punto del proyecto es bastante delicado, ya que este influirá de una forma considerable a nuestro futuro modelo, por lo tanto debemos de tener en cuenta lo siguiente.

- 1. El conjunto de datos determina la capacidad de generalización del modelo:** Si el conjunto de datos de entrenamiento es demasiado pequeño o no es representativo de las diferentes posibilidades de modificación de imágenes, el modelo puede tener dificultades para detectar imágenes modificadas en el mundo real. Un conjunto de datos amplio y variado puede mejorar la capacidad de generalización del modelo y permitir que se ajuste mejor a diferentes situaciones.
- 2. El conjunto de datos influye en la precisión:** Si el conjunto de datos de entrenamiento contiene imágenes mal etiquetadas o ruidosas, el modelo puede ser menos preciso en la detección de imágenes modificadas. Por lo tanto, es importante elegir un conjunto de datos de alta calidad y con etiquetas precisas.
- 3. El conjunto de datos influye en el rendimiento:** Un conjunto de datos grande y variado puede mejorar el rendimiento del modelo, permitiendo una mejor detección de imágenes modificadas y reduciendo el riesgo de sobreajuste.
- 4. El conjunto de datos puede influir en el tipo de modelo que se utiliza:** El conjunto de datos puede influir en la elección del tipo de modelo que se utiliza para la detección de imágenes modificadas. Por ejemplo, si el conjunto de datos contiene una gran cantidad de imágenes con modificaciones sutiles, puede ser más apropiado utilizar un modelo basado en GAN o en aprendizaje por transferencia.

5.1. Dataset elegido

Después de analizar varios conjuntos de datos, he decidido elegir **Casia-2** [6]. Este conjunto de imágenes es ideal para crear un modelo de detección de imágenes modificadas, ya que contiene diversas categorías de imágenes, como **Animales, Arquitectura, Arte, Personas, Interiores, Plantas y Textil**. Además, dentro de cada categoría hay dos subcategorías principales: originales y modificadas. En cada una de estas subcategorías se aplican diferentes tipos de modificaciones, como recortes y fusiones de la misma imagen (**Same**) o de otras imágenes (**Different**).

Aunque este dataset no es perfecto, ya que muestra diversas fuentes de problemas, que pueden incluir lo siguiente:

- 1. Calidad y cantidad del dataset:** El rendimiento del modelo podría mejorarse aún más si se dispusiera de un dataset de mayor calidad y tamaño. La disponibilidad de más ejemplos de entrenamiento, con una mayor diversidad y representatividad de las clases objetivo,

podría ayudar a capturar mejor las variaciones y mejorar la capacidad de generalización del modelo.

2. **Variabilidad en condiciones de captura:** Las condiciones de captura de las imágenes utilizadas en el dataset pueden introducir variabilidades que afectan el rendimiento del modelo. Factores como la **iluminación**, el **ángulo de captura** y las **condiciones ambientales** pueden influir en la precisión de la detección y requerir un análisis más detallado.
3. **Sesgo en el dataset:** Es crucial tener en cuenta la posible presencia de sesgos en el dataset utilizado. El conjunto de datos no representa de manera equitativa todas las clases, lo que resulta en un desequilibrio significativo en las muestras y puede afectar la capacidad de generalización del modelo. Para abordar este problema, se ha utilizado la creación de datos sintéticos, lo cual ha sido fundamental para mitigar los problemas asociados a este sesgo.

Para consultar los ejemplares de cada categoría, tanto las imágenes originales como las imágenes modificadas, puedes referirte a los apéndices A y B, donde se muestran las figuras correspondientes a cada categoría.

Tal como se había comentado este dataset presenta un problema, ya que está algo desequilibrado en cuanto a las categorías, no tenemos la misma cantidad de imágenes para cada categoría. Este problema se puede apreciar en la figura 5.

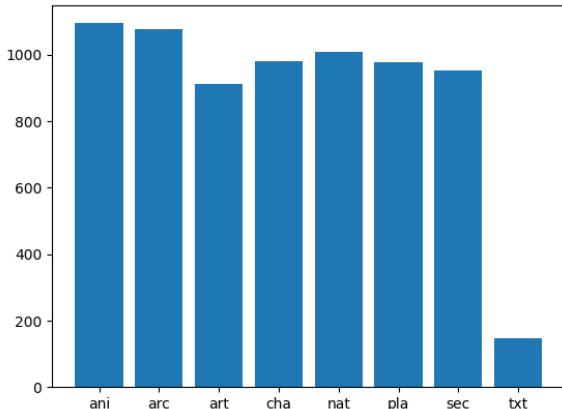


Fig. 5: Cantidad de imágenes por categoría en el conjunto original.

Afortunadamente, se encontró una solución efectiva para abordar este desafío. Se decidió recortar el dataset y limitar cada categoría a solo 600 imágenes. Esta medida permitió lograr un equilibrio en el conjunto de datos. Aunque esto implicó tener menos muestras por cada categoría, se utilizaron técnicas como el **Incremento de datos** para aumentar artificialmente el número de imágenes. Desafortunadamente, la categoría de **Textil** contaba con muy pocas imágenes, por lo que se optó por eliminarla para simplificar el problema. El resultado se muestra en la figura 6.

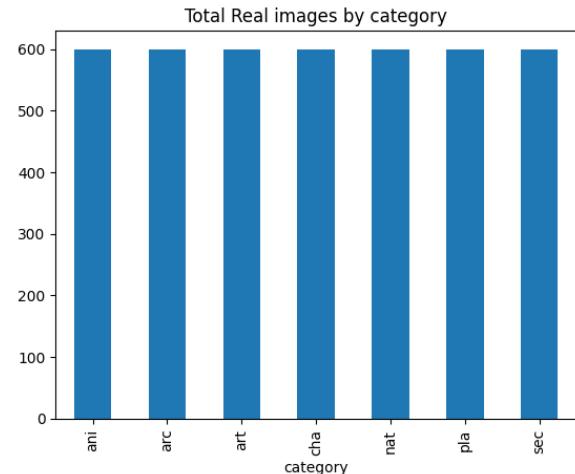


Fig. 6: Cantidad de imágenes por categoría en el conjunto original después de corregirlo.

El mismo problema ocurre en las imágenes de cada categoría en el conjunto modificado, pero en este caso, hay una variable adicional: no todas las categorías tienen la misma cantidad de imágenes con distintas modificaciones. Este desequilibrio se puede apreciar mejor en la figura 7.

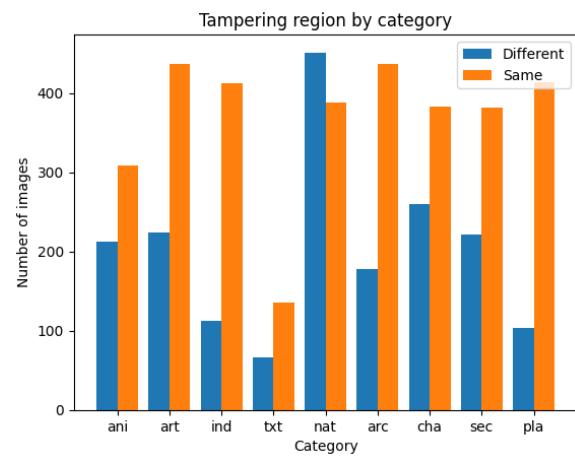


Fig. 7: Cantidad de imágenes por categoría y método del conjunto modificado.

Para abordar este problema, utilizaremos la misma técnica que se empleó con el conjunto de imágenes originales. Sin embargo, en este caso, nuestro objetivo es mantener la misma distribución de datos, es decir, asegurarnos de tener la misma cantidad tanto de imágenes originales como de imágenes modificadas. Para lograrlo, conservaremos 300 imágenes de cada categoría y subcategoría. Por ejemplo, en la categoría de animales, recortaremos 300 imágenes de animales con modificaciones en la misma imagen (subcategoría **same**) y otras 300 imágenes con modificaciones provenientes de otras imágenes (subcategoría **different**). De esta manera, mantendremos un total de 600 imágenes por categoría, al igual que en el conjunto de imágenes originales.

Podemos observar el resultado de esta técnica en la figura 8.

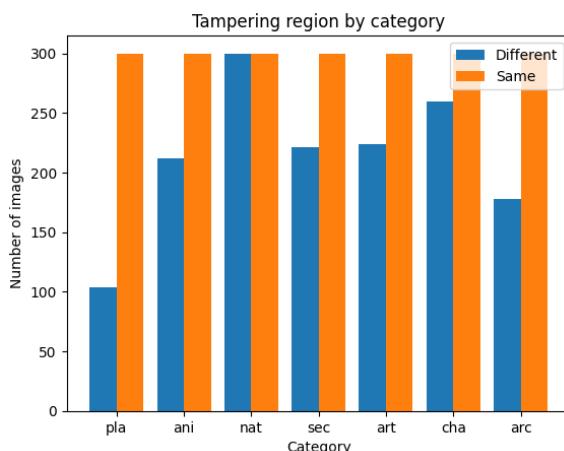


Fig. 8: Cantidad de imágenes por categoría y método del conjunto modificado tras corregirlo.

5.2. Alternativas

Lamentablemente, encontrar conjuntos de datos que cumplan con los requisitos necesarios para desarrollar un modelo robusto ha sido un desafío. La mayoría de los conjuntos de datos disponibles eran de pago o demasiado específicos para nuestra investigación. Por lo tanto, se optó por utilizar el conjunto de datos **Casia-2** [6], que cumplió con nuestras necesidades.

Aunque existe una alternativa al conjunto de datos seleccionado, que es la versión anterior, **Casia-1** [7], se decidió no utilizarla. Esto se debe a que los propios creadores del conjunto de datos indican que la calidad del contenido de esta versión es inferior a la de la versión 2.

5.3. Distribución

El proceso de entrenamiento de un modelo de aprendizaje automático generalmente implica dividir los datos en conjuntos de entrenamiento y prueba para evaluar su rendimiento. En este caso, los datos de entrenamiento se han dividido en un **80 %** para el conjunto de **entrenamiento** y un **20 %** para el conjunto de **prueba**.

Dentro del conjunto de entrenamiento, se ha realizado una subdivisión adicional donde el **20 %** de los datos se ha asignado al conjunto de **validación**. El propósito de tener un conjunto de validación es medir el rendimiento del modelo durante el entrenamiento y ajustar sus hiperparámetros si es necesario.

Una vez que se ha establecido la división entre **entrenamiento**, **validación** y **prueba**, se ha aplicado una técnica llamada **data augmentation** al conjunto de entrenamiento. La **data augmentation** es un proceso en el que se generan muestras adicionales al modificar los datos existentes de manera controlada. Estas modificaciones pueden incluir rotaciones, recortes, cambios en el brillo, entre otros.

6 TÉCNICAS

Durante este apartado se explicarán las técnicas usadas durante todo el trabajo y en que consisten estas.

6.1. Análisis de nivel de error

La técnica de análisis de nivel de error (**Error Level Analysis, ELA**) [8] es una técnica forense que se utiliza para identificar áreas de una imagen que puedan haber sido manipuladas. El principio detrás de esta técnica es que cuando una imagen se comprime y se guarda en un formato con pérdida, como JPEG, la imagen se divide en bloques de píxeles y se aplica un nivel de compresión a cada bloque.

Si una imagen se ha modificado y se ha guardado nuevamente, se espera que el nivel de compresión de las áreas modificadas sea diferente al de las áreas originales. La técnica ELA compara los niveles de compresión en diferentes áreas de la imagen y resalta las áreas donde los niveles de compresión son diferentes, lo que sugiere una posible manipulación.

Podemos observar el resultado que se obtiene al aplicar esta técnica a una imagen modificada en la figura 9

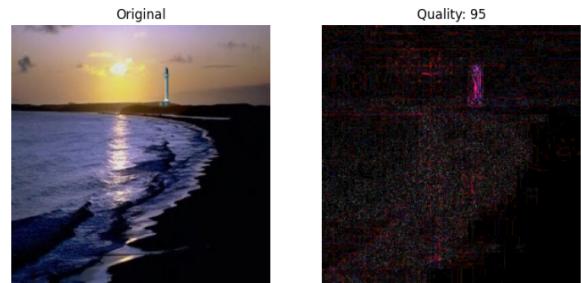


Fig. 9: Aplicando ELA a 95 % a una Imagen modificada.

Como se puede observar en la imagen resultante de aplicar ELA, la modificación realizada en la imagen corresponde a la arquitectura de un faro al fondo de la imagen.

Para lograr este resultado, el primer paso es comprimir la imagen y luego descomprimirla. En el caso de la figura 9, se aplicó una compresión del 95 %. Después de descomprimir, se calcula la diferencia entre la imagen descomprimida y la imagen original, aplicando un factor de 15 para resaltar aún más las posibles modificaciones.

Puedes consultar la sección C de los anexos para observar cómo la calidad de la imagen varía y afecta su apariencia.

6.2. Wavelet

La técnica de **Wavelet** [9] es una herramienta matemática ampliamente utilizada en el procesamiento de señales y el análisis de imágenes. Su objetivo principal es descomponer una señal o imagen en diferentes componentes de frecuencia, lo que permite un análisis más detallado y eficiente de la información contenida en la imagen.

En el contexto del análisis de imágenes, la transformada de **Wavelet** puede detectar patrones y características relevantes, lo que la hace muy útil en aplicaciones de procesamiento de imágenes y visión por computadora.

Al aplicar la transformada de **Wavelet** a una imagen, se obtienen principalmente tres imágenes que representan

diferentes niveles de detalle: **LH (codificación horizontal)**, **HL (codificación vertical)** y **HH (codificación diagonal)**. Cada una de estas imágenes muestra detalles de alta frecuencia en diferentes orientaciones y direcciones, lo que proporciona información valiosa sobre la estructura y textura de la imagen.

En la figura 10 se muestra un ejemplo de cómo se aplica la transformada de **Wavelet** a una imagen modificada y se obtienen las tres imágenes de detalle correspondientes. Estas imágenes pueden ser utilizadas en diferentes aplicaciones, como la detección de objetos y la eliminación de ruido, entre otras.

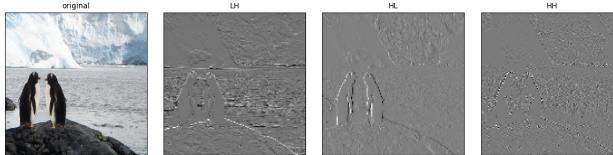


Fig. 10: Aplicando Wavelet a una Imagen modificada (Original - LH - HL - HH).

Para implementar esta técnica empezamos convirtiendo las imágenes a escala de grises utilizando la función **tf.image.rgb_to_grayscale**. Esto es necesario para realizar la transformada wavelet en una única dimensión de intensidad en lugar de en cada canal de color.

Una vez obtendio la escala de grises aplicamos la transformada de wavelet bidimensional utilizando la función **pywt.dwt2**. La wavelet utilizada en este caso es la 'haar', que es una de las wavelets más comunes. Esta transformada descompone la imagen en una aproximación de baja frecuencia (**LL**) y detalles de alta frecuencia en las direcciones horizontal (**LH**), vertical (**HL**) y diagonal (**HH**).

6.3. Luminancia y Crominancia YUV

El espacio de color **YUV** [10] es un sistema de representación de colores que separa la información de luminancia (**Y**) de la información de crominancia (**U** y **V**). La luminancia representa el brillo o la intensidad de la imagen, mientras que la crominancia se refiere a la información de color y tonalidad.

El espacio de color **YUV** se utiliza en aplicaciones de compresión de video y procesamiento de imágenes. Una de las ventajas de este espacio de color es que separa la información de luminancia de la información de crominancia, lo que permite aplicar diferentes técnicas de procesamiento de imágenes y compresión de forma más eficiente.

En cuanto a la detección de imágenes modificadas, el espacio de color **YUV** puede ser útil en determinadas situaciones. Al tener la información de luminancia separada de la crominancia, es posible analizar y comparar los cambios en la luminancia para detectar alteraciones o manipulaciones en la imagen.

A continuacion se muestra en la figura 11 aplicar la tecnica de **YUV** a una imagen e analizando por ejemplo la lu-

minancia de la imagen.



Fig. 11: Obtención de información de la Crominancia (U) de la imagen

Para obtener la crominancia **U** en una imagen, primero se debe de convertir la imagen de formato **RGB** en formato **YUV**.

El proceso de conversión de **RGB** a **YUV** se realiza mediante cálculos matemáticos que implican la combinación lineal de los componentes **R** (rojo), **G** (verde) y **B** (azul) de cada píxel de la imagen. La fórmula para la conversión de **RGB** a **YUV** la podemos encontrar en la figura 12

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ U &= -0.147R - 0.289G + 0.436B \\ V &= 0.615R - 0.515G - 0.100B \end{aligned}$$

Fig. 12: Ecuaciones para calcular la YUV

Una vez que se ha realizado la conversión a **YUV**, la crominancia **U** se obtiene a partir del componente **U** de cada píxel de la imagen.

6.4. Aumento de datos

En el campo del Deep Learning, la disponibilidad de un conjunto de datos lo suficientemente grande y diverso es esencial para entrenar modelos de aprendizaje profundo eficaces. Sin embargo, en este trabajo, al trabajar con el dataset **Casia-2** [6], nos enfrentamos a la limitación de disponer de un conjunto de datos bastante reducido. Además, debido a la necesidad de equilibrar el dataset, se tuvo que recortar aún más las muestras disponibles.

Afortunadamente, existe una técnica que nos permite abordar esta limitación y aumentar nuestro dataset mediante la generación de muestras ficticias. El objetivo al aplicar esta técnica es incrementar nuestro conjunto de datos de forma que las nuevas imágenes generadas no sean demasiado diferentes de las originales, evitando así que nuestro modelo no pueda generalizar correctamente debido a datos muy distintos.

Existen diversas técnicas de aumento de datos, pero en este trabajo se optó por realizar rotaciones tanto **verticales** como **horizontales**, con el objetivo de que este aumento resulte beneficioso para el modelo.

A continuación, en la figura 13, se muestra un ejemplo del aumento de datos aplicado sobre una imagen original.

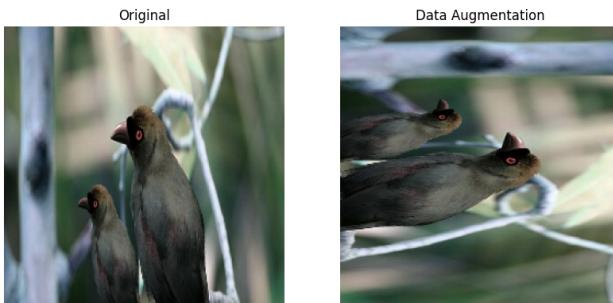


Fig. 13: Aumento de datos

Aunque en la figura 13 se muestra un ejemplo de este aumento de datos, se recomienda consultar la sección D de los anexos para obtener un mayor conocimiento sobre las muestras generadas artificialmente.

6.5. Parada Anticipada

Tambien conocido como **Early Stop** [11] es una técnica utilizada en el entrenamiento de modelos de aprendizaje automático, especialmente en el campo del Deep Learning. Su objetivo principal es evitar el sobreajuste (**overfitting**) [12] del modelo durante el entrenamiento y permitir que el modelo generalice mejor en datos nuevos.

La idea detrás del **Early Stop** es monitorear el rendimiento del modelo en el conjunto de validación a medida que avanza el entrenamiento. Si se observa que el rendimiento en el conjunto de validación deja de mejorar o incluso empeora después de cierto número de iteraciones, se detiene el entrenamiento prematuramente. Esto se hace antes de que el modelo tenga la oportunidad de sobreajustarse al conjunto de entrenamiento y memorizar los datos de entrenamiento en lugar de aprender patrones generales.

El criterio de detención temprana se basa en una métrica de evaluación, como el **Accuracy** o el **Loss**, que se calcula en el conjunto de validación. Si la métrica no mejora o empeora después de un cierto número de iteraciones (*umbral*), se considera que el modelo ha alcanzado su punto óptimo y se detiene el entrenamiento.

Sin embargo, es importante encontrar el momento adecuado para aplicar el **Early Stop**. Si se detiene el entrenamiento muy temprano, es decir, antes de que el modelo haya tenido suficiente tiempo para aprender patrones complejos en los datos, puede resultar en un modelo **subóptimo**. En este caso, el modelo no habrá tenido la oportunidad de converger hacia una solución óptima y su rendimiento en datos nuevos será deficiente.

Por otro lado, si se aplica el **Early Stop** demasiado tarde, es decir, después de que el modelo ya se haya **sobreajustado** al conjunto de entrenamiento, el modelo seguirá entrenándose innecesariamente y es probable que el rendimiento en datos nuevos se vea afectado **negativamente**. Esto se debe a que el modelo habrá memorizado los datos de entrenamiento en lugar de capturar patrones generales, lo que limita su capacidad para generalizar correctamente.

Además, para ilustrar el impacto del **Early Stop** en el entrenamiento de modelos, se presentará la figura 14 que muestra cómo la implementación de esta técnica puede ser beneficiosa en términos de tiempo y recursos durante el entrenamiento, tal como indica el articulo **Early Stopping in Practice** [13].

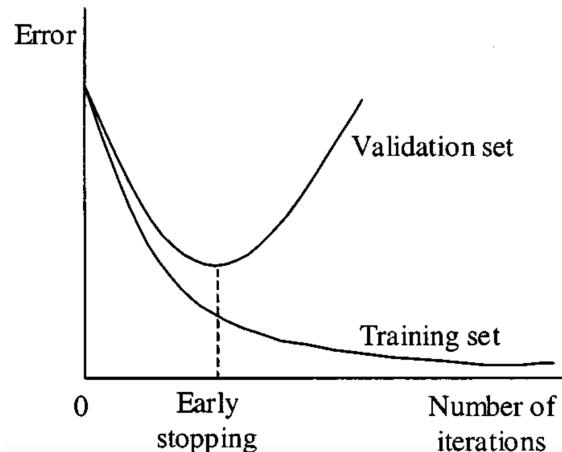


Fig. 14: Ejemplo del impacto del Early Stop

6.6. Transferencia de aprendizaje

La técnica de transferencia de aprendizaje (**Transfer learning**) [14] es un enfoque utilizado en el campo del aprendizaje automático y la inteligencia artificial, que consiste en aprovechar el conocimiento previamente adquirido por una red neuronal entrenada con millones de datos para una tarea específica y aplicarlo a una nueva tarea relacionada. En lugar de entrenar un modelo desde cero en la nueva tarea, se utiliza un modelo preentrenado como punto de partida.

Al aplicar transferencia de aprendizaje, se toma un modelo preentrenado y se ajusta a una tarea más específica, en este caso, la **detección de imágenes manipuladas**. Durante el proceso de ajuste o *fine-tuning*, se modifican y entranan únicamente las capas finales del modelo, mientras que las capas iniciales que ya han aprendido características generales se mantienen fijas.

La transferencia de aprendizaje es beneficiosa porque permite aprovechar el conocimiento previo de la red neuronal, lo cual puede acelerar el proceso de entrenamiento y mejorar el rendimiento en la nueva tarea. En lugar de empezar desde cero, se parte de un punto donde el modelo ya ha aprendido representaciones útiles de las imágenes, lo que puede ser especialmente beneficioso cuando se cuenta con un conjunto de datos de entrenamiento más limitado.

6.7. Grad-CAM

Cuando una Red neuronal CNN realiza una clasificación de imagen, se aplica una serie de operaciones para extraer características significativas de la imagen. Estas características se propagan a través de la red y se utilizan para hacer predicciones. El Grad-CAM se utiliza para

comprender qué partes de la imagen contribuyen más a la predicción realizada por la red.

El **Grad-CAM** [15] utiliza el gradiente de la salida deseada con respecto a las activaciones de una capa convolucional en la red. Básicamente, calcula cómo cambia la salida deseada si se realizan pequeñas modificaciones en las activaciones de la capa convolucional. Estos gradientes se utilizan para ponderar las activaciones de la capa convolucional, resaltando las regiones que tienen una influencia significativa en la predicción.

Por lo tanto, aplicar esta técnica nos proporciona información valiosa, ya que nos indica la razón del porqué esa clasificación, lo cual puede ser el primer paso para tomar una decisión final sobre si una imagen ha sido editada o no.

En la figura 15 se muestra el resultado de aplicar el **Grad-CAM** de la librería de **Keras** [16].



Fig. 15: Ejemplo del resultado del Grad-CAM

7 RESULTADOS

Antes de analizar en detalle los resultados obtenidos en este trabajo, es importante mencionar que se utilizó una GPU NVIDIA RTX 3060 y una CPU AMD Ryzen 5600X, lo que permitió reducir significativamente el tiempo de entrenamiento y evaluación de los modelos de deep learning empleados.

7.1. Resultados Generales

Comenzaremos presentando una tabla con los mejores cinco resultados obtenidos, y luego entraremos en detalle sobre cada uno de ellos y explicaremos las razones detrás de sus resultados. Si observamos la tabla 1, podremos apreciar las distintas métricas obtenidas durante la evaluación.

La elección del mejor al peor se ha basado en el equilibrio de estas tres métricas principalmente: **Accuracy**, **Loss** y **F1-Score**. Aunque existen otras métricas disponibles, se han considerado estas como las más relevantes. Si desea consultar más métricas, puede referirse al apartado de los anexos E.

Modelo	Accuracy	Loss	F1-Score
ENB1_v2_E	0.92	0.2	0.92
ENB3_E	0.92	0.2	0.92
XC_E	0.90	0.55	0.92
MN_E	0.91	0.22	0.91
MN_YUV	0.92	0.23	0.91

Table 1: RESULTADOS GENERALES

7.2. Mejor Resultado

Se ha desarrollado el modelo personalizado denominado **CM_E** (*Custom Model*), que se refiere a un modelo adaptado para facilitar el uso y evitar dependencias para el usuario final. Este enfoque personalizado encapsula todas las funciones de preprocessamiento de imágenes y funciones adicionales en una única clase. De esta manera, el usuario solo necesita proporcionar una imagen al modelo, y este se encargará internamente de adaptarla al formato de entrada requerido.

Durante la fase de experimentación, se llevaron a cabo diversas pruebas que involucraron diferentes combinaciones de técnicas de preprocessamiento de imágenes y modelos de clasificación. El objetivo era determinar la configuración más efectiva para identificar imágenes manipuladas. Después de evaluar los resultados, se encontró que la combinación de la técnica de preprocessamiento conocida como **Error Level Analysis (ELA)** y el modelo base de clasificación **EfficientNetB1** [17] ofreció un rendimiento superior a otras opciones.

El modelo **CM_E** que usa la arquitectura y configuración del modelo **ENB1.v2_E** que se muestra en la figura 1, que consta aproximadamente de **7,888,008** parámetros, de los cuales **7,825,953** son entrenables, experimentó una modificación en la entrada original. Se aumentó su tamaño de **224x224x3** a **256x256x3**. Esta modificación permite que el modelo tenga acceso a más información de píxeles, lo cual mejora su precisión sin comprometer en gran medida su eficiencia. Para comprender más a fondo la estructura interna de este modelo, se puede consultar la sección de los anexos F. Estos anexos proporcionan una visión detallada y exhaustiva de la estructura interna del modelo, permitiendo un mayor entendimiento de su funcionamiento y características específicas.

Es importante destacar que el modelo **CM_E** aprovecha el *transfer learning* del modelo preentrenado **EfficientNetB1**. Esta técnica permite aprovechar los conocimientos previos de una red neuronal ya entrenada y adaptarla a una tarea específica, acelerando el proceso de entrenamiento y mejorando el rendimiento del modelo.

El enfoque **CM_E** se destaca por su naturaleza personalizada, la combinación efectiva de la técnica de preprocessamiento **ELA** y el modelo base **EfficientNetB1**, y el aprovechamiento del *transfer learning* para lograr resultados prometedores en la identificación de imágenes manipuladas.

Además, gracias a este *Custom Model*, se ha integrado

de manera fácil y eficiente la técnica de **Grad-CAM**. Esto dota al modelo de una potente capacidad de clasificación de imágenes modificadas y de indicación de posibles zonas modificadas. Los resultados de aplicar esta técnica se pueden observar en la figura 16.

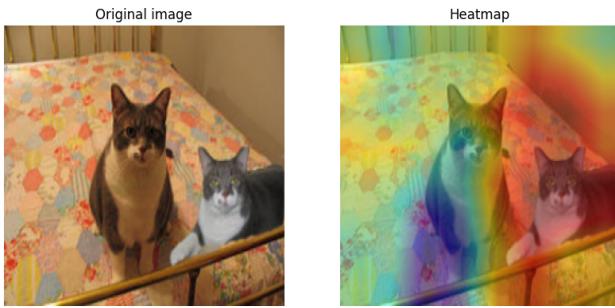


Fig. 16: Grad-CAM del modelo ENB1_v2_E.

Dada la naturaleza de la imagen, se puede deducir que la modificación se encuentra en uno de los dos gatos que aparecen en la imagen, más específicamente en el gato ubicado más a la derecha. Sin embargo, para imágenes que no sean tan claras, esta técnica puede ser de gran ayuda.

Para brindar más información que respalde el éxito de este modelo en comparación con otros, se puede observar en la figura 17 la matriz de confusión. La matriz de confusión proporciona una representación visual de las predicciones del modelo en términos de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Este análisis detallado permite evaluar la capacidad del modelo para distinguir entre imágenes manipuladas y no manipuladas, y así comprender mejor su desempeño y fiabilidad.

		Confusion Matrix	
		Original	Manipulated
Original	Original	1333	86
	Manipulated	27	834

Fig. 17: Matriz de confusión del modelo CM_E.

En la sección de los anexos G se ofrece un análisis detallado de la evolución del modelo a medida que avanzaban las épocas de entrenamiento. En estos anexos se proporciona una descripción minuciosa de cómo el modelo se fue transformando y mejorando a lo largo del proceso de entrenamiento.

7.3. Otros Resultados

Como alternativa, disponemos del modelo **ENB3_E**, que aplica la misma técnica que nuestro mejor modelo *Transfer learning*, pero con la diferencia de que utiliza como arquitectura base el modelo **EfficientNetB3**. Este modelo tiene un total de **11,177,264** parámetros, de los cuales **11,089,961** son entrenables.

Si observamos la tabla 1, podemos notar que las métricas son idénticas, excepto por otras que podemos consultar en los anexos E en el cual sobretodo en el tiempo de inferencia es algo superior.

Aun así, al observar la figura 18, podemos notar cómo este modelo nos indica las modificaciones en la imagen del gato.

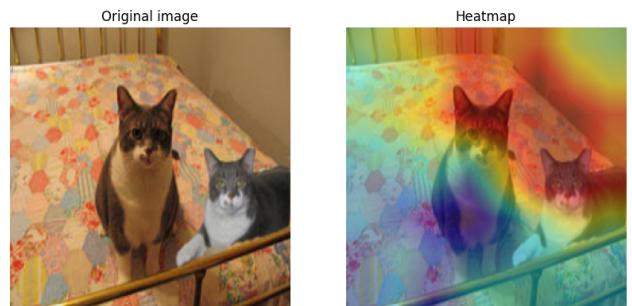


Fig. 18: Grad-CAM del modelo ENB3_E.

Aunque el resultado obtenido no es malo podemos observar que el *ENB1_v2_E* tiene un desempeño ligeramente superior.

8 CONCLUSIONES

En este estudio, se ha demostrado la eficacia del uso de Deep Learning en la detección de modificaciones en imágenes. El modelo propuesto, basado en redes neuronales convolucionales, ha logrado resultados precisos y ha demostrado ser una herramienta valiosa en el análisis forense y la verificación de autenticidad de imágenes.

El tratamiento cuidadoso del dataset utilizado, asegurando un equilibrio adecuado de muestras por clase, ha sido fundamental para garantizar la eficacia del modelo. La capacidad del modelo para capturar patrones sutiles y complejos en las imágenes ha permitido una detección más precisa y robusta.

Además, se ha observado la robustez del modelo frente a diferentes tipos de manipulaciones, lo que demuestra su versatilidad y aplicabilidad en diversos escenarios de análisis forense digital.

Sin embargo, existen desafíos y oportunidades para futuras investigaciones. Se pueden explorar mejoras en el modelo mediante el uso de técnicas avanzadas de arquitectura de redes neuronales y el análisis de imágenes generadas por IA. Además, se pueden abordar limitaciones específicas, como el sesgo en el dataset y la detección de imágenes ficticias generadas por otras IA.

9 LIMITACIONES

A pesar de los resultados obtenidos en este trabajo, es importante reconocer que existen limitaciones que pueden afectar la interpretación y generalización de los hallazgos. A continuación, se detallan algunas de las principales limitaciones y posibles áreas de mejora:

- Limitaciones en imágenes generadas por IA:** Existe una limitación particular en la detección de imágenes generadas completamente por otras IA, como **Dalle-2** [18], **Midjourney** [19], entre otras. Estas imágenes ficticias pueden presentar desafíos adicionales, ya que a nivel de píxeles pueden no mostrar inconsistencias evidentes y podrían ser clasificadas erróneamente como *Originales*. Explorar técnicas avanzadas de detección de imágenes generadas podría ser una dirección prometedora para superar esta limitación.
- Escalabilidad y eficiencia:** A medida que el tamaño del dataset y la complejidad del modelo aumentan, puede surgir la necesidad de abordar cuestiones de escalabilidad y eficiencia computacional. Explorar técnicas de optimización, como el uso de hardware especializado o la implementación de estrategias de entrenamiento distribuido, puede ser crucial para mejorar el rendimiento y la eficiencia del modelo.

REFERENCIAS

- [1] Adobe. Adobe Analytics, <https://acortar.link/2Qtak9>, 2012. 2
- [2] Raúl Álvarez. Adobe, el creador de Photoshop, está desarrollando software para detectar imágenes manipuladas... con Photoshop, <https://acortar.link/PeLl6m>, 2018. 2
- [3] Sheng-Yu Wang, Oliver Wang, Andrew Owens, Richard Zhang, Alexei A. Efros. Detecting Photoshop-ed Faces by Scripting Photoshop. *ICCV*, 2019. 2
- [4] Thanh Thi Nguyen, Quoc Viet Hung Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, Thien Huynh-The, Saeid Nahavandi, Thanh Tam Nguyen, Quoc-Viet Pham, Cuong M. Nguyen. Deep Learning for Deepfakes Creation and Detection: A Survey, *arXiv:1909.11573*, 2022. 2
- [5] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, Matthias Nießner. Face-Forensics++: Learning to Detect Manipulated Facial Images, *arXiv:1901.08971*, 2019. 3
- [6] NPHAT SOVATHANA. casia dataset v2, <https://www.kaggle.com/datasets/sophatvathana/casia-dataset>, 2018. 3
- [7] NPHAT SOVATHANA. casia dataset v1, <https://www.kaggle.com/datasets/sophatvathana/casia-dataset>, 2018. 5
- [8] MarsAnalysisProject. Image Forensics, https://forensics.map-base.info/report_2/index_en.shtml, 2016. 5
- [9] Koushik Chandrasekaran. 2D-Discrete Wavelet Transformation and its applications in Digital Image Processing using MATLAB, <https://acortar.link/cq8jPp>, 2021. 5
- [10] Wikipedia. YUV, <https://en.wikipedia.org/wiki/YUV>, 2004. 6
- [11] Jason Brownlee. Use Early Stopping to Halt the Training of Neural Networks At the Right Time, <https://acortar.link/w8QGLE>, 2020. 7
- [12] Xue Ying. An Overview of Overfitting and its Solutions, *10.1088/1742-6596/1168/2/022022*, 2019. 7
- [13] B. Chen. Early Stopping in Practice: an example with Keras and TensorFlow 2.0, <https://acortar.link/ccPyU1>, 2020. 7
- [14] Tokio School. Analizamos qué es y para qué se usa el Transfer Learning en el Deep Learning, <https://www.tokioschool.com/noticias/transfer-learning/>, 2022. 7
- [15] DataScientest. ¿Qué es el método Grad-CAM?, <https://datascientest.com/es/que-es-el-metodo-grad-cam>, 2022. 8
- [16] fchollet. Grad-CAM class activation visualization, https://keras.io/examples/vision/grad_cam/, 2020. 8
- [17] Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, *arXiv:1905.11946*, 2019. 8
- [18] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents, *arXiv:2204.06125*, 2022. 10
- [19] Jonas Oppenlaender. The Creativity of Text-to-Image Generation, *arXiv:2206.02904*, 2022. 10

ANEXOS

TABLA DE CONTENIDO

- A Ejemplares imagenes originales
- B Ejemplares imagenes modificadas
- C Efecto de la calidad en la funcion ELA
- D Resultado tras aplicar aumento de datos
- E Otras Metricas
- F Estructura interna del modelo
- G Metricas durante el entrenamiento del modelo

A EJEMPLARES IMAGENES ORIGINALES

En la figura 19 se muestra una imagen de cada categoría, en la cual tiene la siguiente codificación **ani** representa a la categoría **Animales**, **arc** representa **Arquitectura**, **art** es la categoría **Arte**, **cha** hace referencia a las imágenes de la categoría **Personas**, **nat** como su imagen representa es de la categoría **Naturaleza**, **pla** representa las **Plantas** y **txt** hace referencia a la clase **Textil**.



Fig. 19: Ejemplos de cada categoría en el conjunto original.

B EJEMPLARES IMAGENES MODIFICADAS

Igual que en anexo A pero en este caso en la figura 20 se muestra una imagen de cada categoría, en la cual tiene la siguiente codificación **ani** representa a la categoría **Animales**, **arc** representa **Arquitectura**, **art** es la categoría **Arte**,

cha hace referencia a las imágenes de la categoría **Personas**, **nat** como su imagen representa es de la categoría **Naturaleza**, **pla** representa las **Plantas** y **txt** hace referencia a la clase **Textil**.



Fig. 20: Ejemplos de cada categoría en el conjunto modificado.

C EFECTO DE LA CALIDAD EN LA FUNCION ELA

Cuando se modifica la calidad de compresión de una imagen y luego se aplica el Error Level Analysis (**ELA**), se pueden observar ciertos efectos. Al disminuir la calidad de compresión, es probable que aparezcan áreas con mayor nivel de error en toda la imagen. Estas áreas pueden indicar posibles zonas de edición. Sin embargo, es importante tener en cuenta que al reducir la calidad de compresión, también aumentamos el ruido en la imagen, lo que puede llevar a la aparición de falsos positivos.

Por otro lado, si aumentamos la calidad de la imagen, es posible que se reduzcan las áreas de error, lo que podría ocultar posibles modificaciones en la imagen.

Por tanto, es crucial encontrar un equilibrio en el nivel de calidad de compresión utilizado, ya que esto puede hacer que el modelo sea más robusto al evitar falsos positivos sin ocultar modificaciones reales.

Si observamos la figura 21, podremos visualizar cómo el ruido aumenta a medida que se disminuye la calidad de compresión.

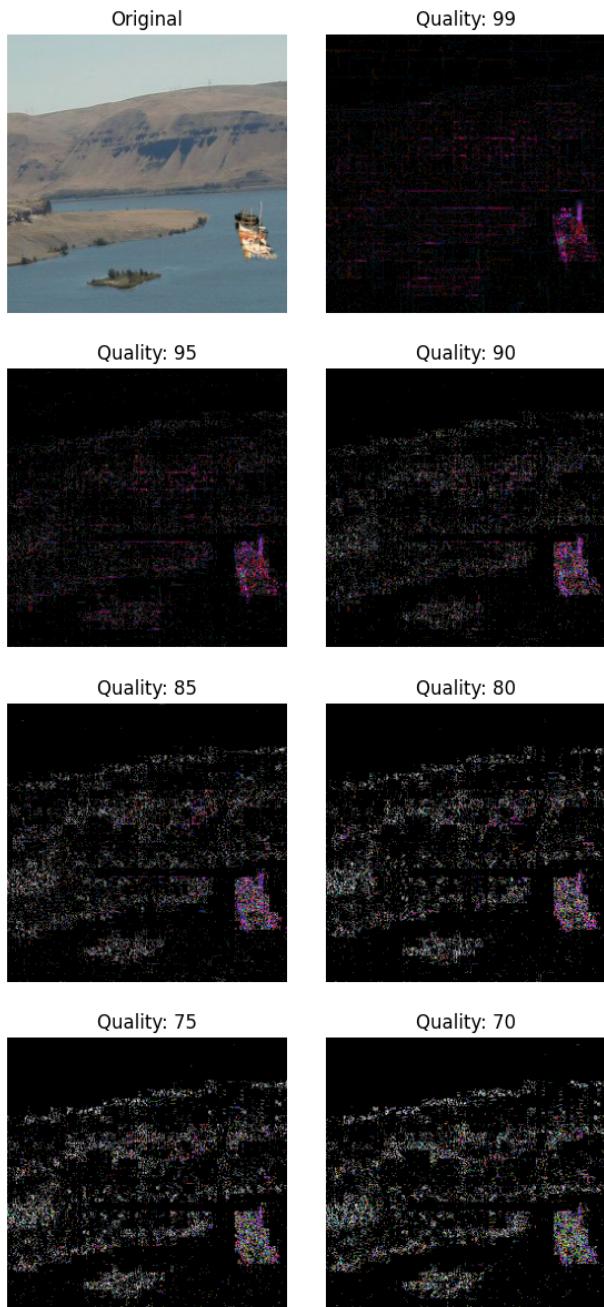


Fig. 21: Ejemplos del efecto de cambiar la calidad.

D RESULTADO TRAS APLICAR AUMENTO DE DATOS

En el estudio realizado, se ha implementado una técnica conocida como **aumento de datos** para mejorar la calidad y la cantidad de los datos utilizados en el análisis. Esta técnica implica generar nuevas muestras a partir de las existentes, aplicando transformaciones específicas.

En particular, se han realizado rotaciones en los ángulos de **90, 180 y 270** grados a las imágenes de muestra. Esto implica girar cada imagen en estas direcciones para obtener versiones adicionales de la misma. Este proceso de rotación ayuda a diversificar el conjunto de datos y proporciona variaciones que pueden ser útiles para entrenar modelos de aprendizaje automático más robustos.

La figura 22 muestra un ejemplo visual de cómo se aplican estas rotaciones en las imágenes de muestra.

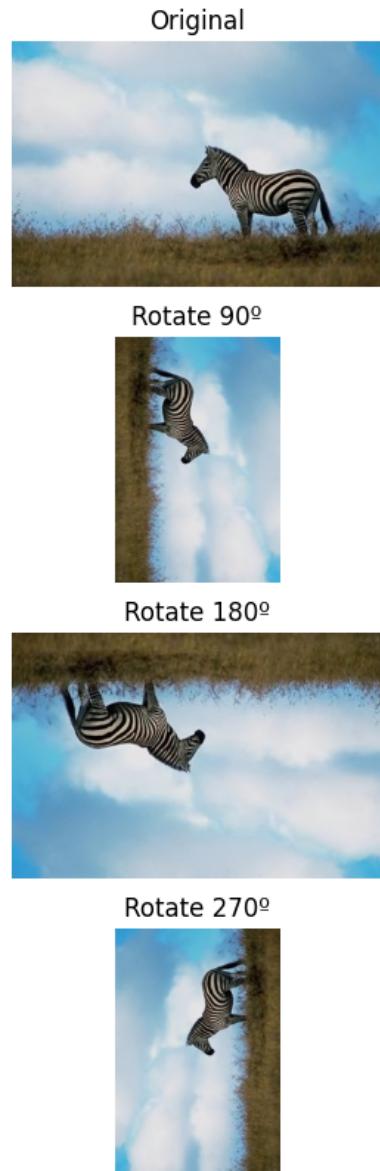


Fig. 22: Posibles resultados tras aplicar el aumento de datos.

E OTRAS METRICAS

En este anexo, se presenta una tabla completa que resume todas las métricas y modelos que se han probado en este estudio. A lo largo de este artículo, se ha llevado a cabo una rigurosa evaluación y comparación de diversas métricas y modelos para abordar el problema planteado. El propósito de esta tabla es proporcionar una visión general de los enfoques que se han explorado, así como los resultados obtenidos.

La tabla 2 muestra una lista exhaustiva de las métricas y modelos que se han considerado en análisis. Se han seleccionado cuidadosamente para cubrir diferentes aspectos y desafíos relacionados con el problema. Cada modelo y métrica ha sido evaluado en términos de su rendimiento y eficacia en la resolución del problema planteado.

Modelo	Epochs	Tiempo por Epoca	Accuracy	Loss	Precision	Recall	AUC	PRC	F1-Score
ENB1_v2_E	13	99s	0.92	0.2	0.92	0.93	0.98	0.98	0.92
ENB3_E	14	126s	0.92	0.2	0.95	0.9	0.98	0.98	0.92
XC_E	12	147s	0.90	0.55	0.90	0.93	0.93	0.94	0.92
MN_E	12	64s	0.91	0.22	0.99	0.84	0.98	0.99	0.91
MN_YUV	33	45s	0.92	0.23	0.90	0.92	0.97	0.97	0.91
ENVB2_E	31	104s	0.89	0.3	0.84	0.98	0.97	0.97	0.91
ENB1_E	31	100s	0.89	0.31	0.85	0.84	0.97	0.97	0.90
XC_YUV	20	130s	0.82	0.83	0.78	0.98	0.91	0.9	0.87
V16_E	15	25s	0.87	0.38	0.80	0.86	0.93	0.87	0.83
ENVB2_E	29	60s	0.78	0.68	0.71	1	0.95	0.94	0.83
ENB1_YUV	18	104s	0.63	0.69	0.62	1	0.49	0.58	0.77
R50_E	7	32s	0.83	0.46	0.81	0.72	0.93	0.88	0.76
XC_W	11	158s	0.62	0.61	0.62	1	0.5	0.62	0.76
V16_W	20	108s	0.62	0.67	0.62	1	0.5	0.62	0.76
ENB1_W	16	125s	0.61	0.67	0.91	1	0.49	0.60	0.76
V16_YUV	14	89s	0.63	0.65	0.65	0.9	0.61	0.7	0.75
Scrath_W	15	42s	0.60	0.68	0.56	1	0.5	0.6	0.75
MN_W	20	67s	0.60	0.68	0.6	1	0.5	0.6	0.75
R50_W	14	90s	0.60	0.68	0.60	1	0.5	0.6	0.75
Scrath_E	12	43s	0.74	1.64	0.94	0.32	0.85	0.82	0.48

Table 2: TABLA DONDE SE MUESTRA TODO LOS MODELOS TESTEADOS JUNTO A SUS METRICAS.

F ESTRUCTURA INTERNA DEL MODELO

A continuación, se presentará la figura 23 que ilustra el resumen (**summary**) del mejor **modelo** obtenido. Esta figura brinda una representación visual de los aspectos más destacados y relevantes del modelo, resumidos de manera concisa y clara. A través de esta imagen, se podrán apreciar de forma visual las principales **características y arquitectura** del modelo. Este resumen visual proporciona una visión general del mejor modelo alcanzado, facilitando su comprensión y evaluación de manera rápida y accesible.

Layer (type)	Output Shape	Param #	Connected to
input_10 (InputLayer)	[None, 256, 256, 3 0]	0	[]
rescaling_1 (Rescaling)	(None, 256, 256, 3) 0	0	['input_10[0][0]']
normalization_1 (Normalization)	(None, 256, 256, 3) 7	7	['rescaling_1[0][0]']
tf.math.truediv_1 (TFOpLambda)	(None, 256, 256, 3) 0	0	['normalization_1[0][0]']
stem_conv_pad (ZeroPadding2D)	(None, 257, 257, 3) 0	0	['tf.math.truediv_1[0][0]']
stem_conv (Conv2D)	(None, 128, 128, 32 864)	864	['stem_conv_pad[0][0]']
stem_bn (BatchNormalization)	(None, 128, 128, 32 128)	128	['stem_conv[0][0]']
stem_activation (Activation)	(None, 128, 128, 32 0)	0	['stem_bn[0][0]']
...			
Total params:	7,888,008		
Trainable params:	7,825,953		
Non-trainable params:	62,055		

Fig. 23: Summary del Modelo.

G METRICAS DURANTE EL ENTRENAMIENTO DEL MODELO

Gracias a la técnica de **EarlyStop**, los modelos han sido capaces de generalizar y obtener buenos resultados en un tiempo de entrenamiento razonable.

Si observamos la figura 24 podemos observar como usando un preprocesamiento **ELA** junto a la arquitectura que nos proporciona el modelo de **EfficientNetB1** se han obtenido las siguientes metricas.

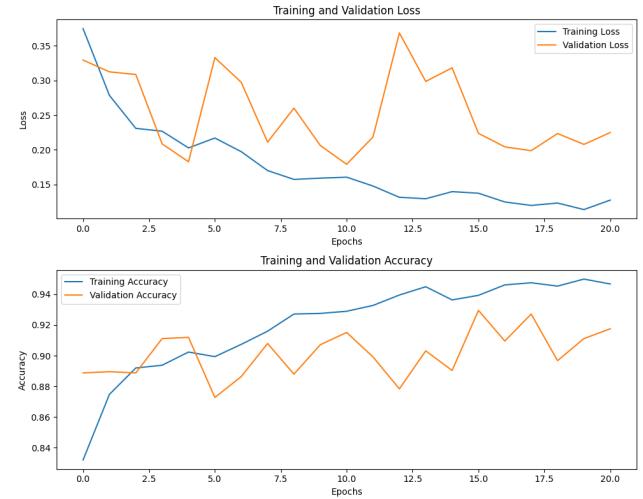


Fig. 24: Metricas del Custom Model con preprocesamiento ELA