Técnica de Detección de Copia y Empalme en Imágenes y Vídeos Basada en Redes Neuronales

– –

Image and Video Copy-Move Forward Detection Technique Based on Neural Networks

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2019–2020

Isauro López Cortegano

Directores

Luis Javier García Villalba
Ana Lucila Sandoval Orozco

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Septiembre de 2020

Agradecimientos

A mi familia, por el esfuerzo que pone en darme oportunidades para formarme profesionalmente y por el apoyo que siempre me ofrece.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Lista de Acrónimos

AI	*Artificial Intelligence*
Cb	*Chroma-Red*
CMF	*Copy Move Forward*
Cr	*Chroma-Blue*
DAE	*Deep Auto Encoder*
DCT	*Transformada Discreta del Coseno*
DL	*Deep Learning*
DWT	*Transformada Discreta de Wavelet*
FN	*Falso Negativo*
FP	*Falso Positivo*
GOP	*Group Of Pictures*
GT	*Ground Truth*
IMM	*Imagen Manipulada*
IMO	*Imagen Original*
KF	*Key-Frame*
MC	*Matriz De Confusión*
ML	*Machine Learning*
RGB	*Red, Green, Blue*

SRM *Modelo Espacial Enriquecido*

SVM *Máquina de Soporte Vectorial*

TN *Negativo*

WLD *Descriptor Local de Weber*

Y *Luminance*

YCrCb *Luminance, Chroma-Blue, Chroma-Red*

Abstract

Videos have become a widely used media file by society to spread news in all kind of areas. The technological development of mobile devices and the popularization of web platforms and social networks have promoted that a large part of digital videos are recorded using hand from mobile devices. In addition, technological progress allow you its distribution and also their manipulation.

Therefore forensic analysis of images and videos of mobile devices gains special importance. This research work proposses an artificial intelligence algorithm based on the techniques used to detect the insertion and deletion of objects in images and videos files. The technique combines of the extraction of characteristics from the multimedia file with the use of symmetric neural networks. In order to evaluate the efficiency of the proposed technique, experiments were carried out using a public data set.

Keywords: Forensics Analysis, Digital Videos, Artificial Intelligence, Copy Move Forward, Deep Autoencoder, Identification, Decomposition in Blocks, Wavelet Transform.

Resumen

Los vídeos se han convertido en un tipo de contenido ampliamente usado por la sociedad en todo tipo de ámbitos. El desarrollo tecnológico de los dispositivos móviles y la popularización de las plataformas web y redes sociales, han favorecido a que gran parte de los vídeos digitales provengan de dispositivos móviles. Además, el avance tecnológico nos sólo nos permite grabar y difundir su contenido, sino también la edición y manipulación intencionada de los mismos.

Las técnicas de análisis forense de imágenes y vídeos de dispositivos móviles cobran pues, especial importancia. En este Trabajo Fin de Grado se propone un algoritmo que utiliza técnicas de inteligencia artificial para detectar la inserción y borrado de objetos en imágenes y vídeos. La técnica combina la extracción de características del contenido del fichero multimedia con el uso de redes neuronales simétricas. Para evaluar la eficiencia de la técnica propuesta se realizaron experimentos utilizando con conjunto de datos públicos.

Palabras clave: Análisis Forense, Vídeos Digitales, Inteligencia Artificial, Copiar-Pegar, Deep Autoencoder, Identificación, Descomposición en Bloques, Transformada Wavelet.

Capítulo 1

Introducción

1.1. Motivación

El desarrollo científico y tecnológico de este último siglo ha cambiado drásticamente nuestra sociedad, posibilitando la automatización de procesos mediante el uso de computadores y dispositivos electrónicos. La mecanización de industrias y tareas administrativas está profundamente ligado con el auge económico, social y tecnológico de nuestra sociedad. Este auge ha favorecido a que en países desarrollados, la mayoría de la población tenga acceso a esta tecnología con facilidad.

Desde sus orígenes, Internet, no ha dejado de crecer; se construyó inicialmente como una red de computadores para conectar comunidades científicas de Estados Unidos, Inglaterra y Francia. En la actualidad Internet se ha convertido en una red de redes, en una red descentralizada de alcance global que nos ofrece multitud de recursos y servicios, como pueden ser: acceder a plataformas oficiales de noticias y telecomunicación a través de páginas web o acceder en las «redes sociales». Entendemos por red social todo aquel servicio o aplicación que nos permite crear un usuario y perfil propio desde el que podemos subir publicaciones, imágenes, vídeos, etc. a nuestro perfil, crear una lista de contactos u amigos digital, compartir y visualizar el contenido que han subido otros contactos. Ejemplos de ello pueden ser: Facebook, Twitter o Instagram.

Internet ha pasado a formar parte de la vida de muchas personas [Hay01]. El acceso a la información es prácticamente inmediato gracias al uso de las nuevas tecnologías; tanto los periódicos y páginas oficiales de los medios de comunicación así como las redes sociales, se han convertido en un medio de difusión de información sin precedente, siendo casi instantáneo y de libre acceso en su mayoría.

Si se considera únicamente a la población que cuenta con un dispositivo móvil; en la actualidad, 7 de cada 10 personas en el mundo tiene un teléfono móvil. De estas 7 personas aproximadamente el 71 % de ellos, es decir 5 de cada 10 personas en el mundo, tiene redes sociales [Kem20].

Con frecuencia, los vídeos que no tienen signos o rastros de edición evidentes son considerados como parte de la verdad al ser hechos reales capturados por dispositivos electrónicos. Sin embargo, con el desarrollo de la tecnología han surgido herramientas potentes y sofisticadas que facilitan de una manera impresionante la edición de vídeos digitales, incluso para quienes no tienen conocimientos técnicos o especializados en el área [GKWB07].

Cuando una gran cantidad de vídeos editados y falsificados aparecen en las redes sociales y medios de difusión, no hay duda de que tendrán un efecto adverso significativo en la estabilidad de la sociedad. Por lo tanto, el estudio forense de vídeos digitales se ha convertido en un tema de investigación muy importante.

1.2. Contexto

Este Trabajo Fin de Grado ha sido realizado dentro del Grupo de Análisis, Seguridad y Sistemas (Grupo GASS, `https://gass.ucm.es/`, Grupo 910623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades del proyecto de investigación THEIA (Techniques for Integrity and Authentication of Multimedia Files of Mobile Devices) con referencia FEI-EU-19-04.

1.3. Objeto de la Investigación

Cuando hablamos de manipulación en vídeos, hacemos referencia a aquellos vídeos que han sido manipulados con el objetivo de engañar alterando la realidad de los hechos que reflejan.

El continuo desarrollo de la tecnología facilita cada vez más la elaboración de vídeos así como la edición de los mismos, por lo tanto es necesario elaborar herramientas que sean capaces de identificar manipulaciones. Estas manipulaciones dejan su huella en los distintos fotogramas del vídeo [SM16], basándonos en las investigaciones realizadas estos últimos años sobre la detección de *Copy Move Forward* (CMF) en imágenes y vídeos, se propone un algoritmo de *Artificial Intelligence* (AI) que permita identificar este tipo de manipulaciones en un vídeo.

1.4. Plan de Trabajo

El desarrollo de este trabajo se ha realizado en tres fases siguiendo una metodología circular basada en el método científico:

1. **Investigación**: El objetivo de esta fase fue la de adquirir el conocimiento técnico necesario sobre el dominio del problema a resolver. Lo que permitiría hacer una lluvia de ideas e hipótesis sobre el algoritmo a desarrollar. Para ello se realizaron reuniones durante los primeros meses del proyecto con el objetivo de contextualizar el problema y resolver las dudas que fueron surgiendo, no solamente con la parte de investigación si no también sobre la tecnología a emplear y el desarrollo con redes neuronales.

2. **Desarrollo**: En esta fase el objetivo principal ha sido implementar el diseño planteado en la fase anterior. Inicialmente no se disponía de un nivel de conocimiento adecuado para abordar el problema, puesto que no he cursado la asignatura de Aprendizaje Automático, me era necesario aprender a usar e implementar redes neuronales ya que mis conocimientos eran puramente teóricos. Por ello participé en conferencias y talleres que, junto con la información y tutoriales que encontrados en Internet, me permitieron adquirir este conocimiento y desarrollar el algoritmo propuesto en este trabajo. Los desarrollos se han realizado con el lenguaje de programación *Python* y con la ayuda de librerías como *Tensor Flow* y *OpenCV*. Inicialmente el algoritmo fue desarrollado con *PyCharm* pero fue posteriormente fue adaptado para poder ejecutar en la plataforma *Google Colab*.

3. **Experimentación**: El objetivo de esta fase es la evaluación del algoritmo diseñado y el posterior análisis de los resultados obtenidos. Estas fases de experimentación también permitieron detectar errores de desarrollo y mejorar la implementación del algoritmo propuesto. Los conjuntos de entrenamiento y de evaluación se elaboraron a partir de los conjuntos de datos proporcionados por el equipo de GASS.

4. **Documentación**: La documentación del trabajo se ha tratado como una tarea independiente al ciclo de trabajo mencionado. Esta tarea se inició poco después de haber adquirido y leído los primeros artículos que estimamos relevantes para nuestro trabajo, esta tarea inicialmente consistió en el apunte de frases y citas de distintos artículos y libros. Dado que tenía conocimientos muy básicos sobre AI y no conocía el lenguaje *Python*; priorizamos el desarrollo del algoritmo, el desarrollo de la memoria no se inició hasta finalizar la primera fase de experimentación.

En la Tabla 1.1 se resumen las tareas realizadas a lo largo del desarrollo del proyecto. Adicionalmente, se adjunta en la Figura 1.1 un diagrama de Gantt que ilustra las tareas y su temporalidad.

Identificador	Tarea
1	Investigación
1.1	Lectura e Investigación de artículos académicos
1.2	Diseño de la solución
1.3	Estudio y análisis del overfitting
2	Desarrollo
2.1	Desarrollo del algoritmo en Google Colab
2.2	Implementación de optimizaciones
3	Experimentación
3.1	Extracción de componentes de imágenes y vídeos
3.2	Experimentos con imágenes
3.3	Experimentos con vídeos
3.4	Análisis de los resultados
4	Documentación
4.1	Anotaciones, citas y recopilación de artículos
4.2	Redacción de la memoria

Tabla 1.1: Tareas Realizadas a lo largo del Proyecto

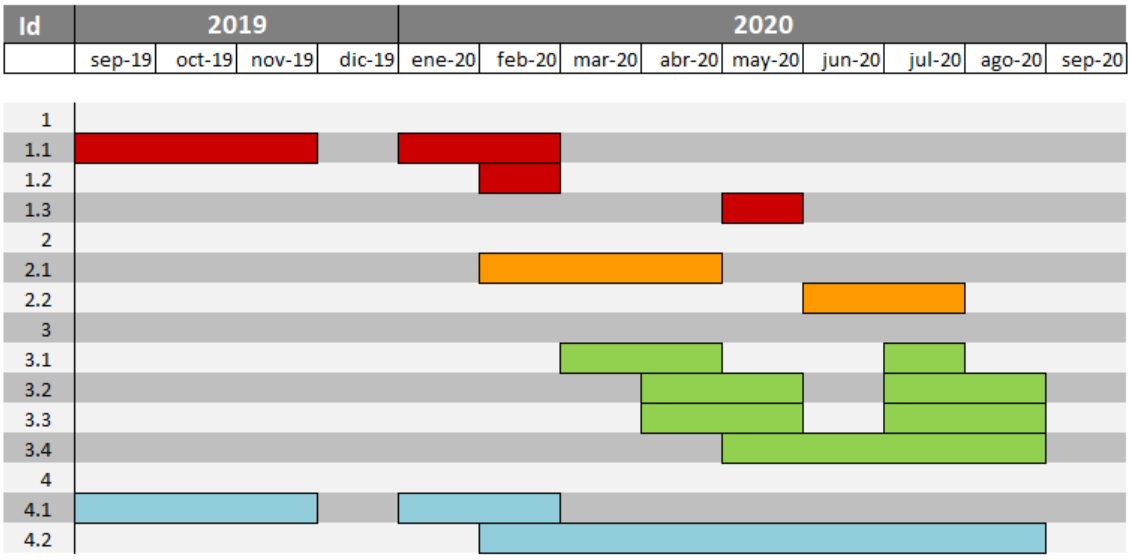


Figura 1.1: Diagrama de Gantt

1.5. Estructura del Trabajo

El resto del trabajo está organizado en siete capítulos con la estructura que se comenta a continuación:

El Capítulo 2 contextualiza e introduce brevemente los conceptos y bases teóricas del trabajo. Concretamente, se describe el funcionamiento de las cámaras digitales y el proceso de digitalización y captura de vídeos e imágenes. A continuación, se presentan los conceptos básicos de la AI. Finalmente, se presentan los distintos tipos de manipulaciones que están relacionadas con la manipulación CMF.

En el Capítulo 3 se presenta brevemente las distintas técnicas de detección de manipulaciones que se emplean en el estudio forense de imágenes y vídeos. Se presenta a continuación, el estado del arte, artículos e investigaciones que han aplicado distintas técnicas para la detectar CMF en imágenes y vídeos.

En el Capítulo 4 se presenta la contribución de este trabajo así como las decisiones y consideraciones que se tomaron a la hora de diseñar el algoritmo.

El Capítulo 5 se describen los experimentos realizados para evaluar la eficacia del algoritmo propuesto en el Capítulo 4. Se analizan los resultados y son posteriormente contrastados con los resultados de los artículos académicos presentados en el Capítulo 3.

El Capítulo 6 concluye este trabajo: resumiendo brevemente la experiencia del proyecto y analizando los resultados del algoritmo propuesto. Se presenta el trabajo futuro y las posibles líneas de investigación.

El Capítulo 7 y 8 son traducciones al inglés del Capítulo 1 y del Capítulo 6.

Capítulo 2

Marco Teórico

2.1. Vídeos Digitales

Todos los vídeos e imágenes son potenciales casos de estudio y la principal fuente de información a la hora de buscar manipulaciones. Por ello es importante no solamente saber extraer información de estos si no también conocer cómo se construyen y se almacenan en la memoria de los dispositivos digitales. Los vídeos suelen estar compuestos por una secuencia de imágenes y una pista de audio. Los dispositivos con capacidad de reproducción digital de vídeos toman esta secuencia de imágenes, como por ejemplo la secuencia ilustrada en la Figura 2.1, y muestran secuencialmente las imágenes contenidas en el vídeo, creando la ilusión de movimiento.

Figura 2.1: Secuencia de Fotogramas de un Vídeo

Hoy en día la inmensa mayoría de los dispositivos móviles cuenta con una cámara digital. Una cámara digital incorpora una serie de lentes, filtros y sensores con la que puede digitalizar la información lumínica que entra por su lente.

La luz entra directamente por la lente y pasa a través de una serie de filtros, de entre los cuales; el filtro de Bayer es un de ellos, este filtro se sitúa sobre un sensor digital de imagen para hacer llegar a cada fotodiodo la información de luminosidad correspondiente de los distintos colores primarios. La tensión de los fotodiodos del sensor de imagen está en constante oscilación. Cuando se toma una foto, el dispositivo móvil lee la tensión de los distintos sensores y almacena esta información en la memoria del dispositivo. La Figura 2.2 esquematiza el proceso descrito anteriormente.

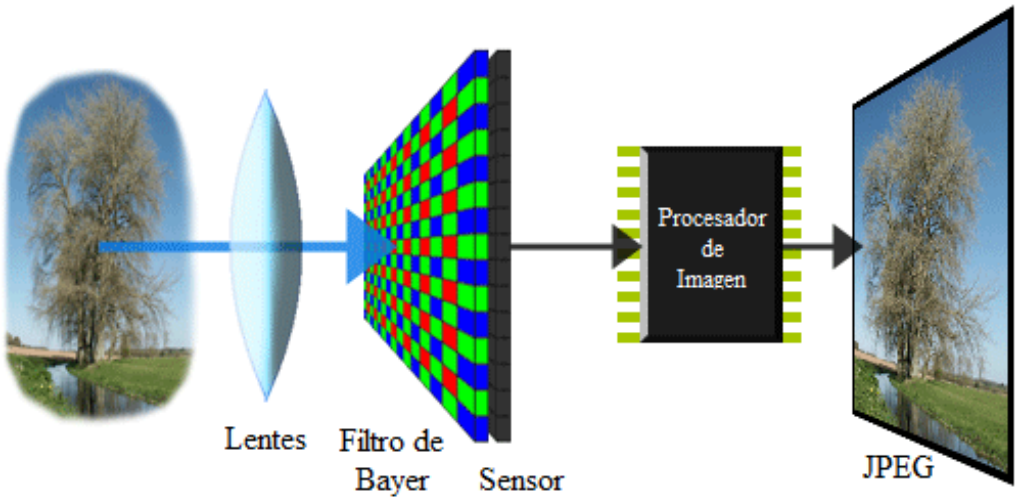


Figura 2.2: Cámara Digital de un Teléfono

Cuando se graba un vídeo, muy frecuentemente la diferencia entre dos fotogramas consecutivos es pequeña. En lugar de almacenar los distintos fotogramas, el dispositivo toma periódicamente un fotograma de referencia y almacena la diferencia entre la información del fotograma actual y el fotograma de referencia. De esta forma se consigue reducir la cantidad de memoria necesaria para almacenar la secuencia de fotogramas.

2.1.1. Compresión

La información capturada por el dispositivo suele comprimirse antes de ser almacenada en la memoria del dispositivo. Estas compresiones de datos son llevadas a cabo por los *codecs* (codificador - decodificador). Su objetivo es reducir el número de datos necesarios para representar las imágenes digitales. Existen dos tipos de *codecs*:

1. **Sin pérdida**: Estos algoritmos comprimen poco los datos y no pierden calidad de

imagen.

2. **Con pérdida**: Estos algoritmos comprimen enormemente los datos del vídeo, de 15 a 30 veces en el caso del MPEG-2, lo que reduce el ancho de banda necesario para transmitir la información considerablemente. No obstante estos algoritmos pierden la calidad original.

El tipo de compresión de un vídeo o imagen cobra especial importancia en este estudio, ya que como se ha mencionado inicialmente, los vídeos son la principal fuente de información. La información que se obtendrá de un caso de estudio se verá afectada por la compresión que haya sufrido.

2.1.2. Grupo de Imágenes

Se conoce como Grupo de Imágenes o estructura GOP, del inglés *Group Of Pictures* (GOP), a un conjunto de fotogramas. Un GOP representa a una sección del vídeo y especifica el orden de los fotogramas contenidos. La Figura 2.3 esquematiza la estructura del GOP. Está compuesto por un fotograma de referencia, conocido también como *Key-Frame* (KF), y por un conjunto de fotogramas de predicción y predicción bi-direccional. Conocidos también como *Fotogramas-I*, *Fotogramas-P* y *Fotogramas-B* respectivamente.

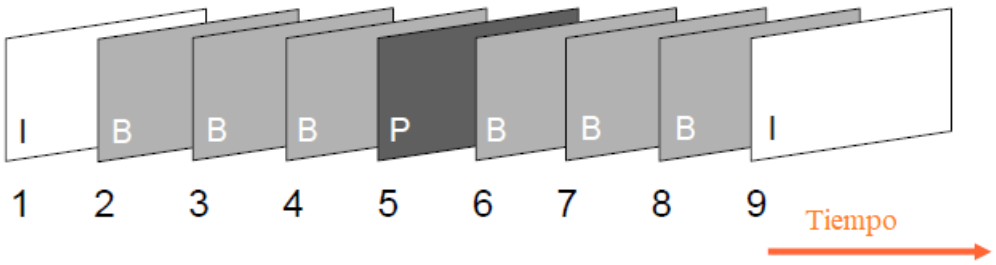


Figura 2.3: Grupo de Imágenes

Cuando se habla de compresión de vídeos, el GOP suele emplearse como la unidad básica de compresión. Puesto que el GOP no requiere de información adicional para poder decodificar la secuencia que contiene. La estructura y número de fotogramas contenido en un GOP depende de los parámetros M y N con los que este es comprimido. El primero de ellos determina la distancia entre el KF y el primer fotograma de tipo P. El segundo determina la distancia entre dos KF (la longitud del GOP). El GOP será por tanto, la estructura de información básica en lo que respecta al tratamiento de los vídeos. Una vez presentados qué son los vídeos, cómo se estructuran y cómo se almacenan, debemos de estudiar qué es la AI y qué nos ofrece para resolver el problema que estudiamos.

2.2. Inteligencia Artificial

La AI es una rama de la computación que trata de diseñar algoritmos y estructuras que imiten las acciones cognitivas de un ser humano, como pueden ser: aprender, razonar, reconocimiento del habla, etc.

En sus orígenes, la idea de la AI suscitó gran cantidad de cuestiones éticas y filosóficas, hay principalmente dos grandes enfoques de la AI: uno más cognitivo y otro más conductista. El primer enfoque defiende la idea de diseñar sistemas pensantes por si mismos, John Haugeland planteaba la idea de elaborar "máquinas con mente, en su amplio sentido literal" (1985). Por otro lado, el enfoque conductista defiende la idea de diseñar sistemas que actuaran como humanos, "¿Cómo lograr que las computadoras realicen tareas que, por el momento, los humanos hacen mejor?" (Delaine Rich & Kevin Knight, 1991).

El objetivo de estos sistemas "inteligentes" es igualar, o superar, dichas capacidades mentales, como puede ser el aprendizaje.

2.2.1. Aprendizaje Automático

En 1952, Arthur L.Samuel elaboró un programa que jugaba a las damas y que era capaz de mejorar con cada partida. Definía la AI como "la capacidad de que tiene un sistema para aprender sin ser programado explícitamente" (1959). Arthur L.Samuel es considerado el pionero del aprendizaje automático, *Machine Learning* (ML) en inglés.

El objetivo de estos algoritmos de ML es, como su propio nombre indica, aprender. Estos algoritmos tienden a perfeccionar su resultado en base a la experiencia que adquieren. La Figura 2.4 ilustra las diferentes entidades de los que se compone un algoritmo de ML, estos algoritmos esperan recibir casos con los que entrenar el modelo o evaluar y predecir respuestas. El modelo contiene la definición del conocimiento necesario para que el algoritmo aprenda a clasificar, discriminar y optimizar las distintas entradas.

Esta rama de la AI está en continuo desarrollo y ha dado lugar a toda una colección de algoritmos de propósitos muy diversos: diagnósticos médicos, clasificación de secuencias de ADN, reconocimiento del habla, etc. La importancia de estos algoritmos radica en la inherente capacidad que tienen para extraer y encontrar, sin estar explícitamente programados para ello, patrones basándose en el análisis de los datos.

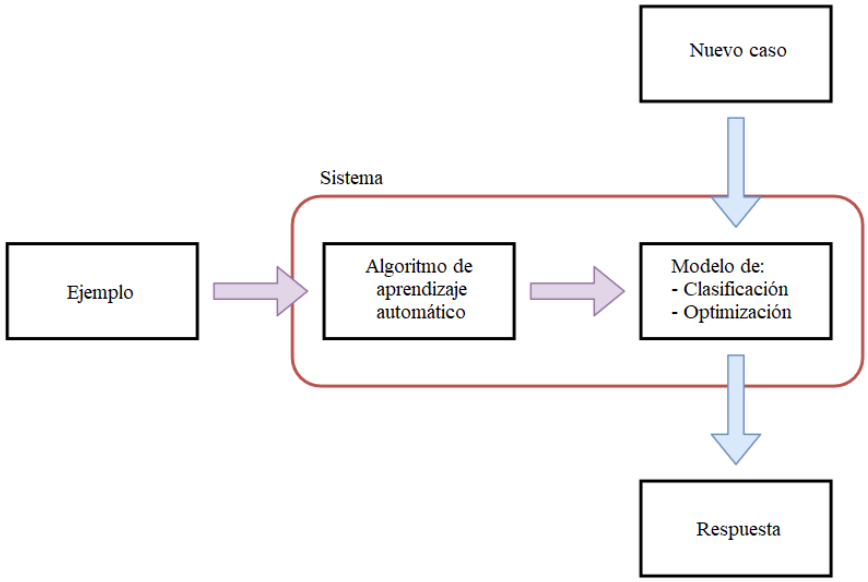


Figura 2.4: Machine Learning - Esquema

2.2.2. Aprendizaje profundo

Poco después Frank Rosenblatt implementó el perceptrón, un clasificador binario que se rige por la siguiente función:

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \text{ is even} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Donde x es un vector de entradas, w un vector de pesos con valores en $\mathbb{R}$ y b es el bias. La Figura 2.5 ilustra de forma esquemática el perceptrón de Frank Rosenblatt, el cual es capaz de abstraer por si solo la funcionalidad de las operaciones lógicas AND y OR.

El origen de este diseño radica en el propio modelo biológico de las neuronas, las cuales cuentan con un número indeterminado de dendritas (entradas en este caso) y un único axón (la salida). La salida del perceptrón, viene determinada por una función de umbral como puede ser la Ecuación 2.1, que determina el valor resultado e impone un límite que se debe sobrepasar antes de propagarse a otro posible perceptrón adyacente. Esta función se conoce como función de activación.

Al igual que la neurona, el perceptrón, por si solo es incapaz de realizar tareas complejas, pero pueden enlazarse entre ellas formando: capas, pipelines y redes de perceptrones, que cuentan con un poder de abstracción más elevado. Estos algoritmos extienden las funcionalidades de los algoritmos de ML y son denominados algoritmos de aprendizaje profundo o *Deep Learning* (DL), en inglés.

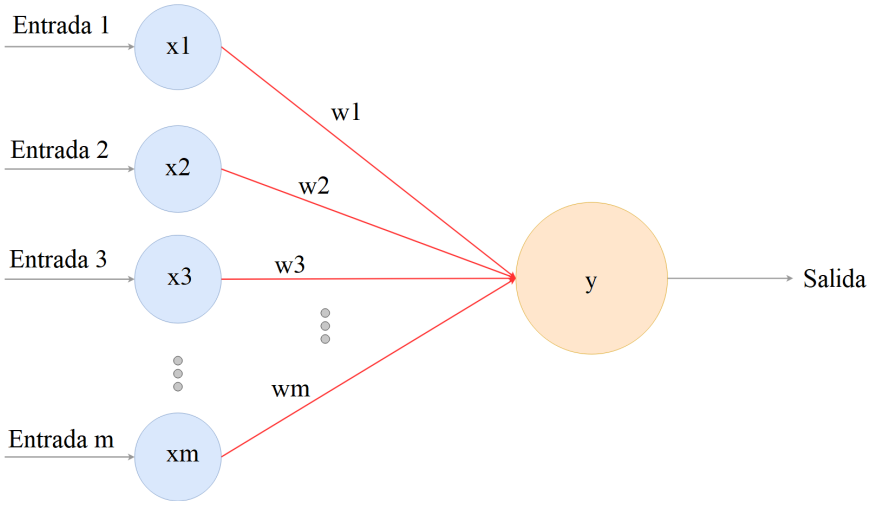


Figura 2.5: Perceptrón de F. Rosenblatt

2.2.2.1. Redes Neuronales

La redes neuronales son estructuras compuestas generalmente por perceptrones inter-conectados. La concepción de una red neuronal está en representar a la red por capas; estas capas están compuestas por perceptrones independientes entre si. La Figura 2.6 ilustra la estructura de una red neuronal compuesta por tres capas de perceptrones, para interconectar las capas de perceptrones se asocian las n-ésimas salidas de la i-ésima capa, con el vector de entrada de cada uno de los perceptrones de la siguiente capa.

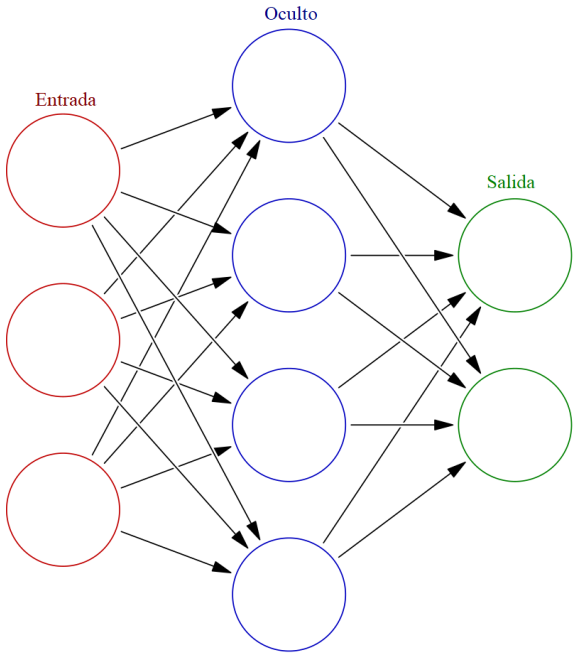


Figura 2.6: Red Multicapa

Estos sistemas en red sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional. En la etapa de entrenamiento de una red neuronal, dependiendo del problema que estemos estudiando, pueden presentarse dos casos:

1. El resultado de los casos de prueba es conocido.
2. El resultado de los casos de prueba es incierto, puesto que no existe un conocimiento a priori.

En función de la naturaleza del problema se opta por emplear algoritmos de entrenamiento supervisados o no supervisados, respectivamente. En este caso de estudio puesto que se puede manipular imágenes y conocer los resultados de estas, se emplearán algoritmos de entrenamiento supervisados.

Para ello se debe definir una función de pérdida o coste que permitirá que la red neuronal cuantifique cuán lejos está su predicción del resultado esperado. La red tomará elementos del conjunto de datos, calculará una predicción del resultado y comparará su salida con la esperada. Las redes neuronales tratan de minimizar el resultado de su función de pérdida; los valores de los pesos de las distintas entradas de las neuronas se van actualizando buscando reducir el valor de la función de pérdida, este proceso se realiza mediante la propagación hacia atrás, denominada también regresión.

No obstante, cuando se diseñaron los casos de prueba y se entrenaron un algoritmo de aprendizaje automático se debe prestar especial atención al sobreajuste, o también conocido como **Overfitting**, del inglés. Esta condición puede presentarse en un sistema sobreentrenado, lo que causa que el modelo responda correctamente a los casos de entrenamiento pero falle al evaluar nuevos casos.

La Figura 2.7 se representa gráficamente el problema del *Overfitting*. Siendo los cuadrados amarillos los casos de prueba y la línea roja, la predicción del modelo, se observa que; en la gráfica de la derecha, la predicción del modelo entrenado mapea los casos de entrenamiento y puede responder a estos con una gran precisión. No obstante, el caso ideal ilustrado en la gráfica de la derecha ilustra un modelo entrenado que ha podido generalizar los casos de prueba de tal forma que el modelo resultante se ajusta mucho mejor a los casos nuevos.

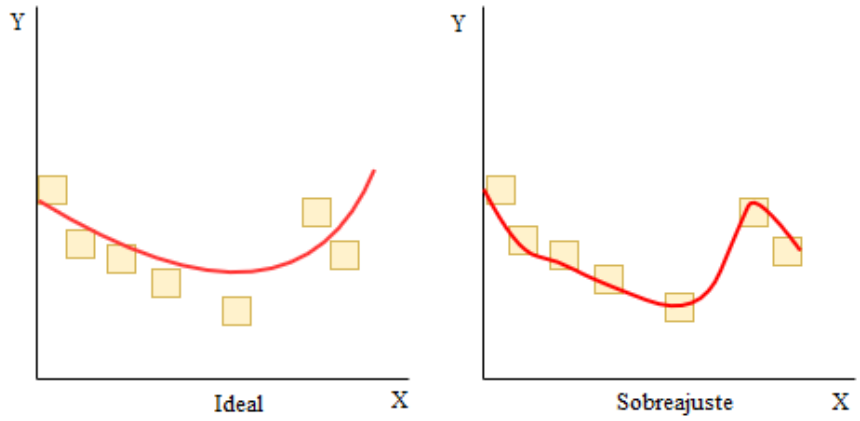


Figura 2.7: Sobreajuste

2.2.2.2. Tipos

Las redes neuronales se agrupan en tipos según la estructura que se les dota. Los tipos de redes neuronales más conocidos son:

1. **Red multicapa**: La red multicapa, se corresponde con el ejemplo presentado en la Figura 2.6, cuenta con una capa de entrada, una de salida y un conjunto de capas intermedias u ocultas que pueden estar totalmente o parcialmente conectadas. Dentro de esta categoría existen diversos modelos de red neuronal según la forma de las capas, se destacan los *Deep Auto Encoder* (DAE) y los clasificadores:

   a) *Codificador Automático*: Un codificador automático o DAE, en inglés, es una red multicapa que se caracteriza por tener una estructura simétrica y estar compuesto de tres partes: un codificador, un *códec* y un decodificador, Figura 2.8.

   Este tipo de red neuronal aprende a copiar la entrada abstrayendo la representación de esta en su capa intermedia.

   b) *Clasificadores*: Los clasificadores son redes multicapa que se caracterizan por tener tantas salidas como propiedades o características de los elementos de entrada que se quieren clasificar (ver Figura 2.9).

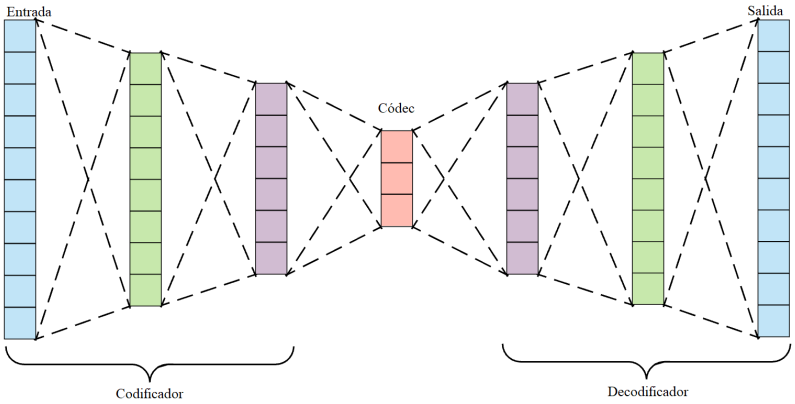


Figura 2.8: Red DAE

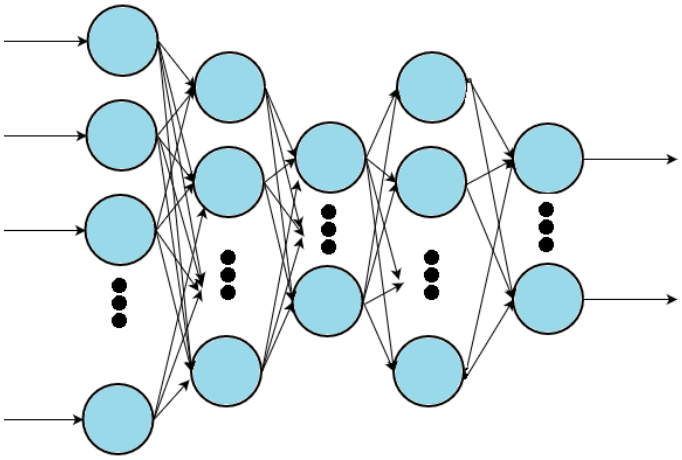


Figura 2.9: Red Clasificadora

2. **Red convolucional**: La red neuronal convolucional tiene una estructura similar a la red multicapa pero se caracteriza por que sus neuronas solamente se unen con un subgrupo de neuronas de la siguiente capa en lugar de unirse con todas. Esto favorece la especialización y reducir su complejidad computacional (ver Figura 2.10).

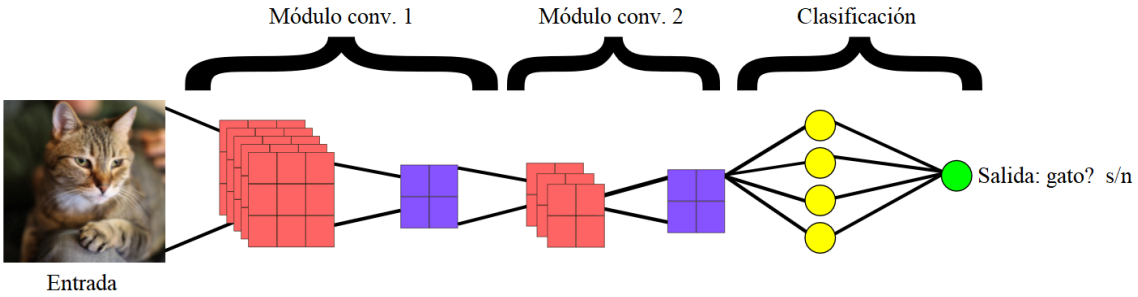


Figura 2.10: Red Convolucional

3. **Red recurrente**: Las redes recurrentes no suelen organizarse por capas, permiten la conexiones arbitraria entre neuronas, pudiendo formar ciclos, permitiendo que la red tenga memoria (ver Figura 2.11).

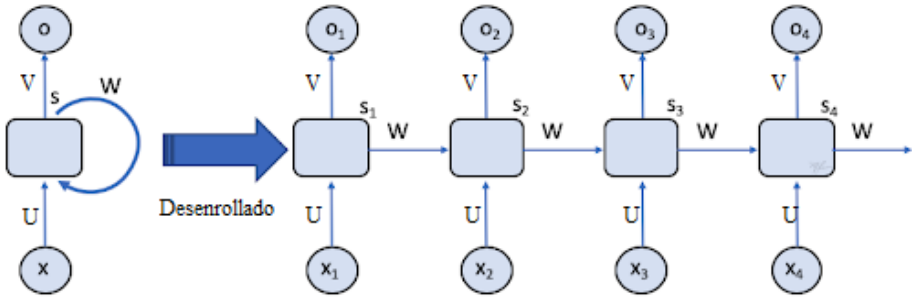


Figura 2.11: Red Recurrente

2.2.2.3. Inteligencia artificial aplicada a la detección de manipulaciones

El campo de la AI está en continuo desarrollo y evolución. Como se presentará en el próximo capítulo, son numerosos los estudios e investigaciones que empiezan a emplear técnicas y algoritmos de DL con el objetivo de resolver los problemas existentes mediante el uso de redes neuronales.

Dada la gran complejidad de los problemas, es cada vez más frecuente el uso de este tipo de algoritmos. Para ello, se realiza un estudio previo sobre qué componentes y características de los fotogramas pueden ser las más significativas a la hora de identificar la manipulación que se desea detectar. Este conjunto de características se convierte en el modelo; el cual primeramente deberá ser entrenado y posteriormente evaluado, analizado y contrastado para medir su efectividad.

2.3. Edición y Manipulación de Vídeos

Para editar un vídeo el delincuente hace uso, normalmente, de programas especializados en edición de vídeo. Estos programas ofrecen multitud de herramientas edición y suelen ser sencillos de usar, como por ejemplo: *Final Cut*, *Sony Vegas*, etc. Cuando una persona manipula el contenido de un vídeo, la intención suele ser una o varias de las siguientes: eliminar u ocultar objetos, agregar objetos o alterar el comportamiento de los objetos que aparecen en una escena.

El objetivo principal del delincuente es, además de alterar el contenido con el objetivo de borrar u ocultar algo, que la modificación que haya realizado sea tan sutil y precisa como para no levantar sospecha y que resulte imperceptible por el ojo humano. Estas

manipulaciones sobre el contenido de los vídeos se pueden agrupar en las siguientes categorías: Las manipulaciones inter-fotograma e intra-fotograma.

2.3.1. Manipulaciones Inter-fotograma

Las manipulaciones inter-fotograma son todas aquellas manipulaciones con las que se logra alterar el flujo original de fotogramas del vídeo. La Figura 2.12 ilustra las manipulaciones más comunes, que consisten en: la inserción, el borrado, la duplicación y el desordenamiento de fotogramas o secuencias del vídeo; pudiendo de esta forma, manipular los acontecimientos que suceden en el mismo.

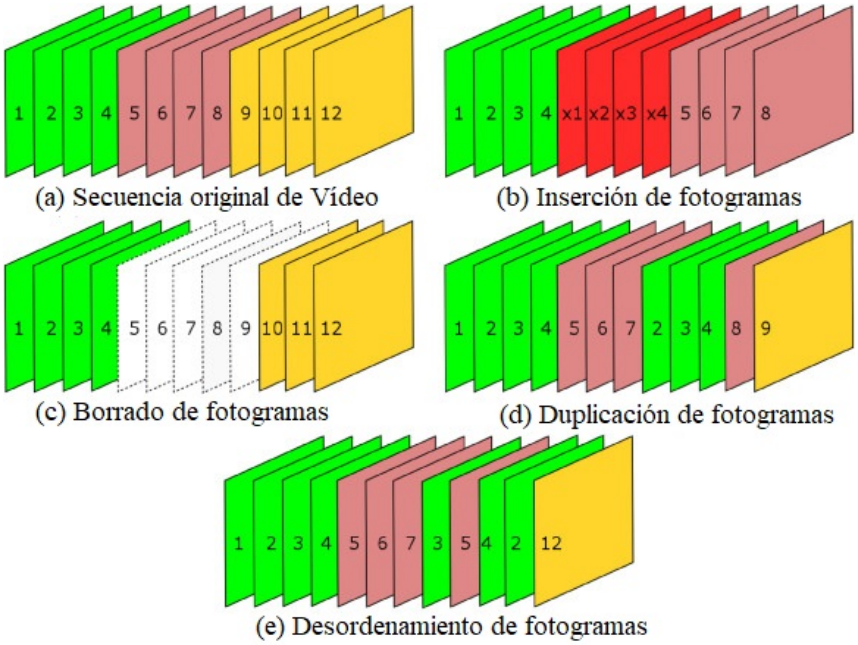


Figura 2.12: Manipulaciones Inter-fotograma

2.3.2. Manipulaciones Intra-fotograma

Todas las manipulaciones intra-fotograma suelen fundamentarse en el copia pega de regiones de imágenes. Una vez se pega el fragmento de la imagen, se aplican herramientas de difuminado y suavizado para disimular las zonas circundantes de la región manipulada. Dentro de las manipulaciones intra-fotograma se diferencian dos campos:

2.3.2.1. Empalme

El empalme, o Splicing en inglés. Este tipo de manipulación se caracteriza por tomar regiones de diferentes imágenes y pegarlas en el fotograma que se desea manipular. La

Figura 2.13 ilustra una manipulación en la que se aplica el empalme de un globo sobre otra imagen.

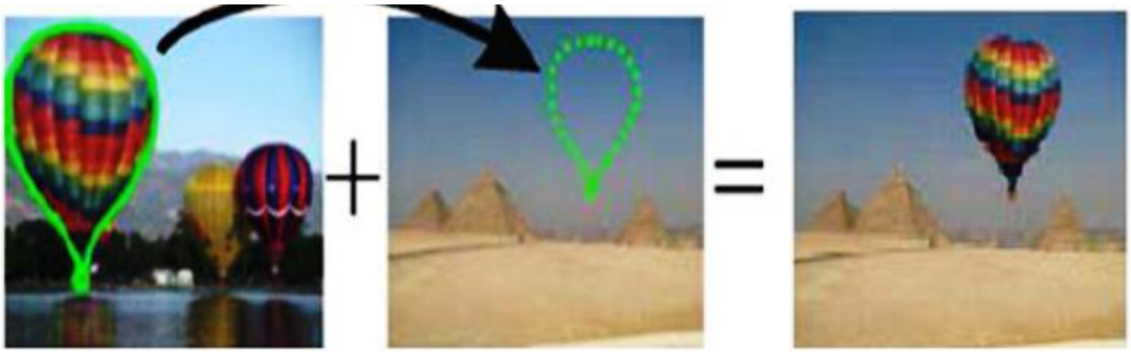


Figura 2.13: Manipulación - Empalme

2.3.2.2. Duplicación de regiones

Este tipo de manipulación se caracteriza por tomar regiones de la propia imagen que está siendo manipulada. Cabe destacar que este tipo de manipulación cuenta a su vez con multitud de variantes, como pueden ser: las rotaciones, deformaciones y traslaciones. En las rotaciones, Figura 2.14, los objetos manipulados presentan giros y/o volteamientos.

Figura 2.14: Manipulación - Rotación

En las deformaciones, Figura 2.15, los objetos manipulados son estirados, agrandados, contraídos, etc.

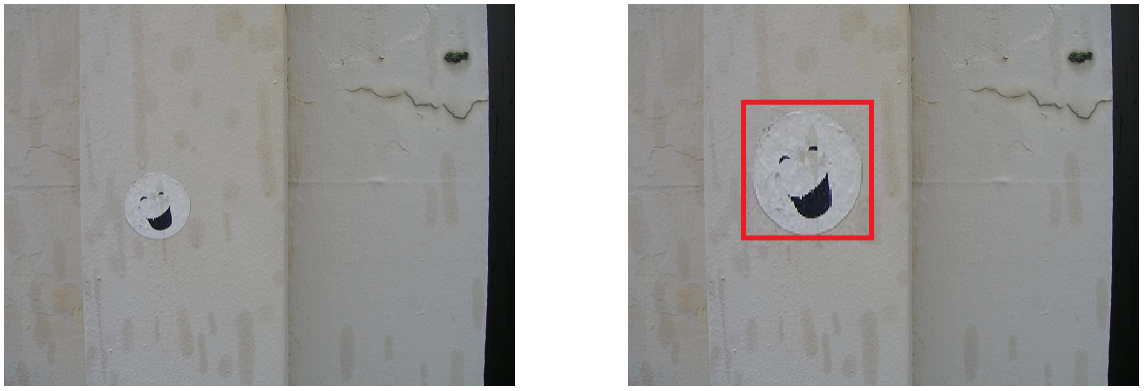


Figura 2.15: Manipulación - Deformación

En las traslaciones, Figura 2.16, los objetos manipulados son trasladados en la escena, pudiendo multiplicarse el número de apariciones de estos en la escena.

Figura 2.16: Manipulación - Traslación

2.3.2.3. Manipulación Copia-Empalme

El copia y empalme o CMF, en inglés, es una técnica de manipulación intra-fotograma basada en la duplicación de regiones.

Se entiende por CMF al proceso de manipulación de una imagen que consiste en copiar una región de la propia imagen y pegarla en otra región de la misma imagen sin ser modificada, como ilustra la Figura 2.17. El estudio de este trabajo estará enfocado a elaborar un algoritmo que permita detectar este tipo de manipulaciones para vídeos e imágenes.

Figura 2.17: Manipulación - CMF

Capítulo 3

Estado del Arte

El estudio forense tiene por objetivo determinar las circunstancias exactas de la comisión de una infracción y la correspondiente identificación de sus autores mediante el empleo del conocimiento científico.

El avance y desarrollo imparable de la tecnología provoca que este campo del Derecho busque resolver los nuevos problemas que surgen. Aplicado al campo de estudio, los vídeos digitales, la Dra. K.Sitara y el Dr. B.M.Mehtre mencionan que "el estudio forense trata de buscar trazas en el contenido de los vídeo con el objetivo de determinar la autenticidad e integridad de los mismos." [SM16].

La Figura 3.1 ilustra los diferentes campos de estudio que abarca el estudio forense de vídeos:

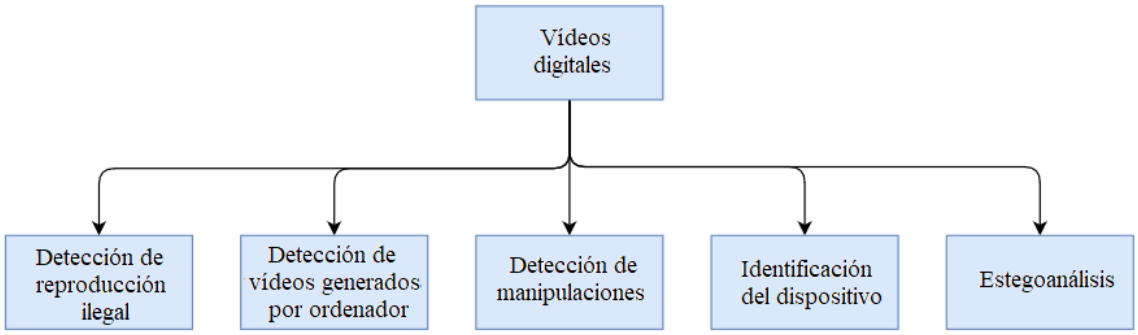


Figura 3.1: Disciplinas Científicas del Estudio Forense de Vídeos Digitales

- **Detección de reproducción ilegal de vídeos**; trata de identificar copias falsificadas de vídeos con el fin de proteger los derechos del autor.
- **Detección de vídeos generados por ordenador**; busca diferenciar vídeos o regiones del mismo que hayan sido generadas por un ordenador.

- **Detección de manipulaciones**; tiene por objetivo encontrar manipulaciones realizadas sobre el propio contenido del vídeo.
- **Identificación del dispositivo**; trata de identificar el dispositivo que grabó el vídeo.
- **Estegoanálisis**; es una disciplina que trata de detectar contenido e información que haya podido ocultarse en un vídeo mediante técnicas estenográficas. La estenografía es el arte de ocultar información en un objeto portador. A diferencia de la criptografía, la estenografía trata que la información oculta sea inadvertida, que el hecho mismo de su existencia y envío sea una incógnita.

Este trabajo centra su atención en estudiar las diferentes técnicas que se emplean para identificar manipulaciones intra-fotograma. Este conjunto de técnicas se agrupan en dos categorías: técnicas activas y técnicas pasivas. Estas técnicas se clasifican en activas o pasivas en función de la naturaleza de la manipulación que estudian:

- Las técnicas activas son aquellas técnicas que tratan de analizar elementos clave presentes en el vídeo con el objetivo de autenticarlo, como pueden ser las marcas de agua o firmas digitales.
- Las técnicas pasivas representan el conjunto de técnicas que tienen por objetivo analizar el contenido de un vídeo: los diferentes fotogramas, el tipo de compresión.. con el objetivo de detectar manipulaciones.

3.1. Técnicas Pasivas

Este conjunto de técnicas son las más relevantes a la hora de analizar la veracidad de un vídeo porque "la autenticidad del vídeo se verifica extrayendo características de este. Cualquier operación de edición dejará huellas en el contenido del vídeo que podrían ser empleadas para verificar su autenticidad" [SM16].

Como ilustra la Figura 3.2, las técnicas pasivas se agrupan en tres categorías principales:

- **Técnicas de detección de manipulaciones inter-fotograma**; Son técnicas que se basan en el análisis de los GOP de un vídeo con el objetivo de detectar el borrado, inserción, duplicación o desorden de los fotogramas que contiene.
- **Detección de múltiple compresión**; "Para manipular un vídeo en formato comprimido, primero tiene que ser descomprimido. Después de editar, el vídeo resultante es almacenado nuevamente en formato comprimido" [SM16].

Los algoritmos empleados para detectar múltiple compresión pueden variar en función del *códec* empleado en la compresión del vídeo, estos estudios suelen basar sus análisis extrayendo las distribuciones de los coeficientes de la transformada del coseno discreta. Esta distribución en condiciones normales suele corresponderse con una distribución Gaussiana o de Laplace, si el vídeo es recomprimido con unos parámetros de compresión diferentes, es probable que la distribución de coeficientes resultante deje de corresponderse con los modelos esperados.

- **Técnicas de detección de manipulaciones intra-fotograma**; Estas suelen basarse en el análisis bloque por bloque de los distintos fotogramas del vídeo, buscando las trazas y extrayendo patrones estadísticos que permitan identificar y/o localizar las regiones manipuladas.

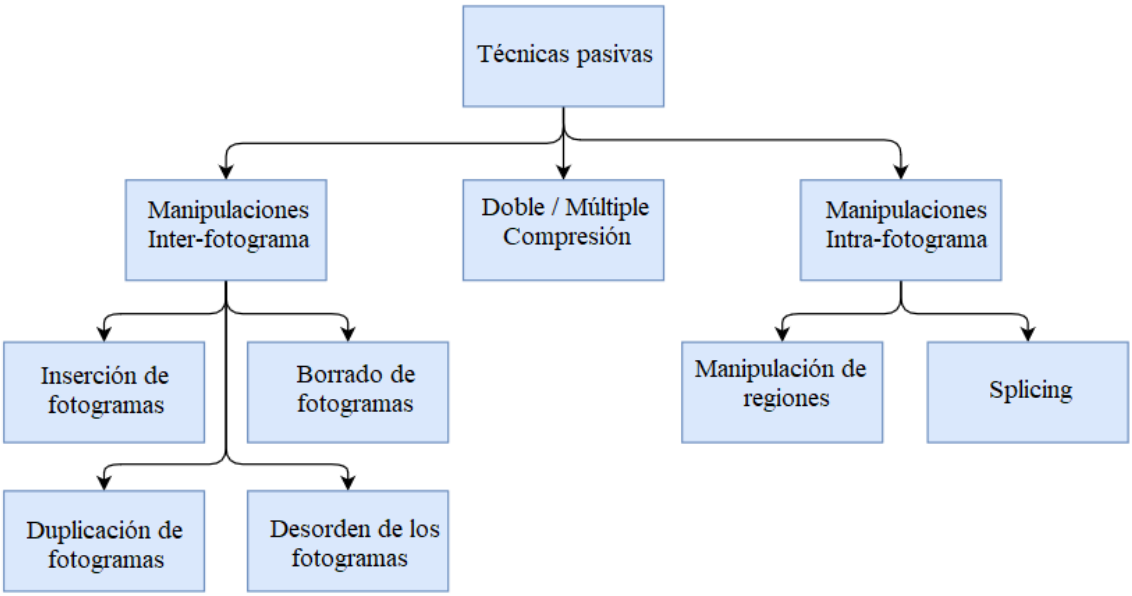


Figura 3.2: Desglose de Técnicas Pasivas Aplicadas a la Detección de Manipulaciones

3.2. Detección de Copia y Empalme

En esta sección del trabajo presentamos el estado del arte, exponemos a continuación un subconjunto de los estudios leídos y detallaremos las diferentes técnicas que en ellos se mencionan, los *datasets* que se emplearon en su experimentación y los resultados y conclusiones que se obtuvieron.

Introduciremos primeramente algunas de las técnicas empleadas para detectar CMF en imágenes y vídeos. Seguidamente, presentaremos trabajos que han aplicado algoritmos de AI para dar solución a problemas relacionados con la detección de manipulaciones en

imágenes y vídeos. Finalmente, analizaremos los artículos expuestos y contrastaremos los resultados que obtuvieron.

3.2.1. Detección de Copia y Empalme en Imágenes

En [SSB10] se presenta un conjunto de técnicas que se aplican para la detección de CMF en imágenes. Las técnicas más relacionadas se presentan a continuación.

3.2.1.1. Búsqueda exhaustiva

La imagen es fragmentada es distintos tamaños y bloques con los que se realiza una búsqueda sobre la propia imagen, tratando de identificar regiones similares, pueden realizarse adicionalmente giros y pequeñas deformaciones sobre el bloque buscado.

"Es la forma más sencilla de detectar CMF" [SSB10]. No obstante, esta técnica solamente es viable cuando las imágenes son pequeñas ya que el coste computacional de este algoritmo es de $(WH)^2$ donde WxH corresponde con el tamaño de la imagen.

3.2.1.2. Transformada Discreta del Coseno

En el estudio [FSJ03] se propone un algoritmo que extrae características de las imágenes empleando la *Transformada Discreta del Coseno* (DCT).

El algoritmo propuesto fragmenta la imagen en bloques y para cada bloque se calculan los coeficientes DCT. Estos coeficientes son almacenados en una matriz A con $(M - B + 1)(N - B + 1)$ filas y BxB columnas, siendo B el tamaño del bloque. Posteriormente, se ordenan los elementos de cada fila lexicográficamente.

Junto con los coeficientes DCT, el algoritmo también tiene en cuenta las coordenadas de los bloques implicados. De esta forma, si el algoritmo se encuentra dos filas consecutivas similares, registra las posiciones de los bloques implicados, como un vector de desplazamiento, en una lista separada e incrementa una cuenta C. Esta cuenta representa la frecuencia de apariciones del vector de desplazamiento v_i.

Finalmente, el algoritmo comprueba todos vectores de desplazamiento. Aquellos cuyo número de apariciones es superior a un umbral k especificado por el usuario es identificado como manipulado y coloreado según el componente *Red, Green, Blue* (RGB) que ha facilitado su identificación, Figura 3.3.

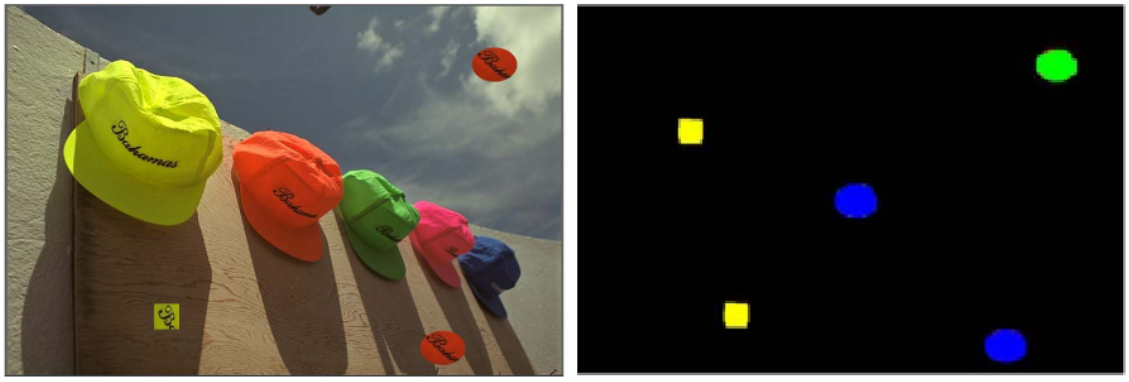


Figura 3.3: Resultados del Estudio: [FSJ03]

3.2.1.3. Transformada Discreta de Wavelet

En el artículo [LWTS07] se presenta un algoritmo similar al anterior que emplea la *Transformada Discreta de Wavelet* (DWT) para extraer las características.

Los resultados experimentales demostraron que este método es efectivo incluso cuando la imagen estaba muy comprimida o procesada los bordes.

3.2.1.4. Coordenadas log-polares

En el artículo [MVP08] se propone un algoritmo basado en la extracción de características empleando la DWT y mapeando los bloques a coordenadas log-polares. El algoritmo propuesto realiza la detección de zonas manipuladas en dos fases. En la primera de ellas se aplica la DWT para reducir la dimensión y complejidad de la imagen y realiza una búsqueda exhaustiva sobre esta. En la segunda, se construye una matriz A con $(M - B + 1)(N - B + 1)$ filas y BxB columnas, siendo B el tamaño del bloque. Un bloque deslizante de tamaño BxB que recorre la imagen píxel por píxel en la que por cada posición del bloque se obtiene su coordenada log-polar correspondiente y los coeficientes Wavelet. Cada fila de la matriz A se corresponde con una posición del bloque deslizante. Esta matriz es ordenada lexicográficamente. Cada fila es comparada con un número n de filas por debajo y encima suya usando la correlación de fase como criterio de similitud. Cuando la correlación entre dos bloques supera un umbral t el algoritmo almacena las referencias de ambos bloques y son marcados como manipulados, Figura 3.4. Este algoritmo dio buenos resultados con los siguientes formatos de imagen: JPEG, BMP y PNG.

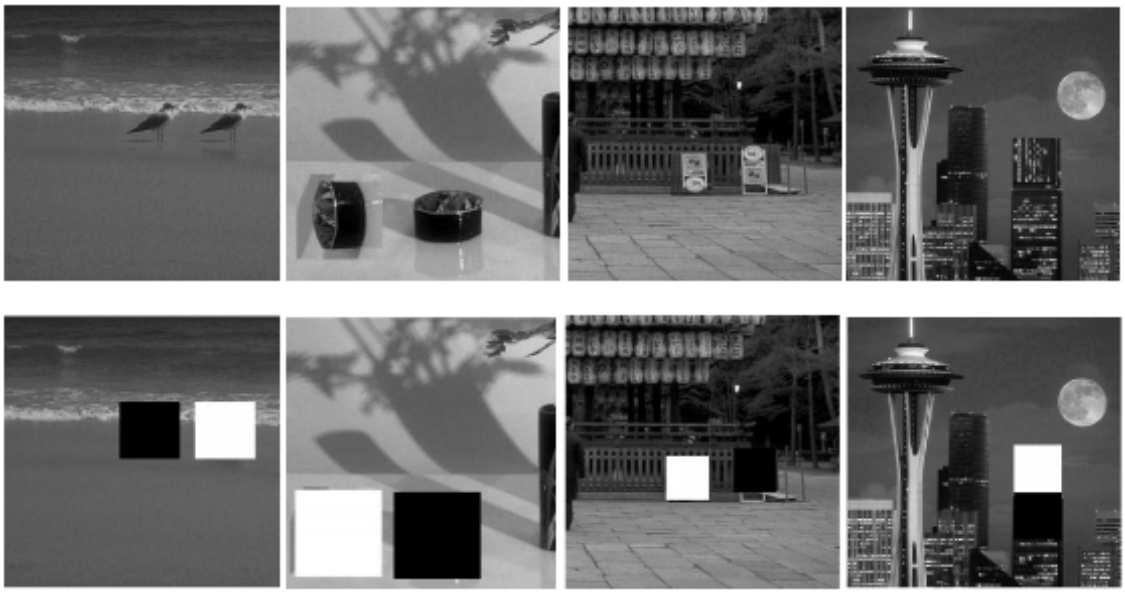


Figura 3.4: Resultados del Estudio: [MVP08]

3.2.1.5. Descriptores Weber

En el artículo [HMS+12] se propone un algoritmo que hace uso de un *Descriptor Local de Weber* (WLD), este descriptor está basado en el hecho de que "la percepción humana de un patrón depende no solo del cambio de un estímulo sino también de la intensidad original del estímulo" [CSH+09]. El algoritmo propuesto toma como entrada imágenes en el espacio de color RGB y aplica un cambio al espacio de color *Luminance, Chroma-Blue, Chroma-Red* (YCrCb), seguidamente extrae las componentes de crominancia para extraer características de la imagen. A continuación calculan y obtienen un conjunto de histogramas con diferentes operadores de variación R y P; siendo R la resolución espacial para el operador y P el recuento de vecinos. Estos histogramas son concatenados entre sí y mediante el uso de una *Máquina de Soporte Vectorial* (SVM) clasifican las imágenes como manipuladas o no manipuladas. Los datos que se emplearon en este estudio correspondían con imágenes manipuladas con CMF. Los resultados de este estudio se presentan en la Tabla 3.1. Este estudio obtuvo una precisión del 91 % extrayendo características sobre las componentes de crominancia del espacio de color YCrCb.

Tabla 3.1: Resultados del Estudio: [HMS+12]

Tipo de Transformación	Accuracy	
	Cr	Cb
Rotación	22.36 %	10 %
Cambio de tamaño	37.77 %	37.77 %
Deformación	80 %	50 %
Nada	91.14 %	90.65 %

3.2.2. Detección de Copia y Empalme en Vídeos

En el artículo [LH13] los autores toman en consideración la métrica SSIM, establecida en el trabajo [WBSS04] sobre las características perceptivas del sistema visual humano, esta métrica mide la similitud a partir de tres componentes estadísticos: La estructura, la luminancia y el contraste.

Cuando se manipula un vídeo, varios fragmentos de un fotograma pueden copiarse en fotogramas posteriores para mantener la temporalidad del objeto que se esté copiando o desplazando. El método propuesto evalúa la similitud de los diferentes fotogramas del vídeo, para ello se toma en consideración que dos fotogramas consecutivos son muy similares, en el momento en el que existan regiones manipuladas este factor de similitud será ligeramente superior o inferior. Esta variación de similitud es considerada como región manipulada si la diferencia de similitud entre fotogramas consecutivos es superior a un umbral t. Este umbral t se calcula en función de la métrica de similitud de los fotogramas adyacentes para una secuencia de vídeo, de esta forma el umbral no es un valor fijo y se ajusta al contenido del vídeo. Los experimentos se llevaron a cabo con vídeos grabados por cámaras digitales y por cámaras de dispositivos móviles en los que se realizaron duplicaciones de fotogramas y copias de regiones. Los resultados demostraron la eficacia del método propuesto pudiendo alcanzar tasas del 99.7 % de precisión.

3.2.3. AI Aplicada a la Detección de Copia y Empalme

En el artículo [RJ16] se propone un método haciendo uso de una red convolucional para la extracción de características. En el método propuesto se realiza un pre-entrenamiento de la red neuronal entrenando todos aquellos bloques marcados como manipulados y entrenando aleatoriamente la misma cantidad de bloques no manipulados. A continuación la red pre-entrenada es empleada para extraer las características de los distintos bloques de la imagen. Este conjunto de características es posteriormente fusionado y empleando por una SVM para detectar falsificaciones. La inicialización de la red convolucional no sigue

una estrategia aleatoria, los pesos de la primera capa se inicializan con unos valores que obtienen haciendo uso de un *Modelo Espacial Enriquecido* (SRM), este modelo es empleado en estudios de esteganálisis para obtener estadísticas de alto orden recopiladas a partir de residuos de ruido de una imagen, que favorecen a la rápida convergencia del modelo. Los experimentos de este artículo se realizaron con los *dataset* CASIA v.1 y CASIA v.2. Los resultados de este estudio se presentan en la Tabla 3.2. Los resultados obtenidos muestran la diferencia de precisión que obtuvo el método propuesto realizando una pre-inicialización de los pesos haciendo uso del SRM y la inicialización pseudo-aleatoria *Xavier*.

Tabla 3.2: Resultados del Estudio: [RJ16]

Dataset	**Tam. bloque**	**Accuracy**	
		SRM-CNN	**Xavier-CNN**
CASIA v1.0	64	98.04 %	87.91 %
CASIA v2.0	64	97.77 %	97.42 %

En el artículo [ZGLWT16] se propone un algoritmo de DL basado en la extracción de componentes con la DWT y el uso de un DAE como red neuronal. El método propuesto transforma las imágenes al espacio de color YCrCb, las cuales son posteriormente fragmentadas en bloques de tamaño B, se calcula a continuación por cada bloque su desviación estándar, la media y la suma total de sus componentes. Adicionalmente, se extraen las componentes verticales, horizontales y diagonales aplicando la DWT de nivel 3 y por cada una se extraen nuevamente estas medidas. Las componentes resultantes corresponden con el modelo que será entrenado por el DAE. Los experimentos de este estudio se realizaron con los *dataset* CASIA v1.0, entrenando imágenes manipuladas con CMF y una imagen de *Ground Truth* (GT) que resaltaba la zona manipulada en blanco sobre un fondo negro. Los resultados del estudio se presentan en la Tabla 3.3.

Tabla 3.3: Resultados del Estudio [ZGLWT16]

	JPEG	**TIFF**
Fall-out	7.09 %	4.39 %
Precisión	59.43 %	80.65 %
Accuracy	87.51 %	81.91 %

En el artículo [DCGV17] los autores proponen un método para detectar CMF haciendo uso de un DAE. Este estudio el algoritmo propuesto recorre los fotogramas del vídeo con un tamaño de bloque de $128x128$. Para la extracción de características hacen uso de un detector de anomalías presentado en el artículo [Coz16] el cual contiene un modelo que ha

sido entrenado con características relacionadas con el ruido de imágenes que capturan los distintos dispositivos móviles. El valor residual de un punto espacial se calcula como:

$$r_{ij} = f_{i,\,j\text{-}1} - 3f_{i,\,j} + 3f_{i,\,j+1} - f_{i,\,j+2} \tag{3.1}$$

Donde f y r son la imagen de origen y la imagen residual respectivamente, siendo i y j coordenadas espaciales. Para la detección de empalmes se analizan secuencias de cuatro píxeles sucesivos. Obtienen así, un conjunto de histogramas de co-ocurrencias que posteriormente se vectorizarán, obteniendo el vector de características final.

Para su entrenamiento y evaluación se emplearon vídeos de corta duración extraídos del *dataset* `www.grip.unina.it`. Los resultados experimentales demostraron ser efectivos, especialmente si el vídeo presentaba algún tipo de compresión, Figura 3.5.

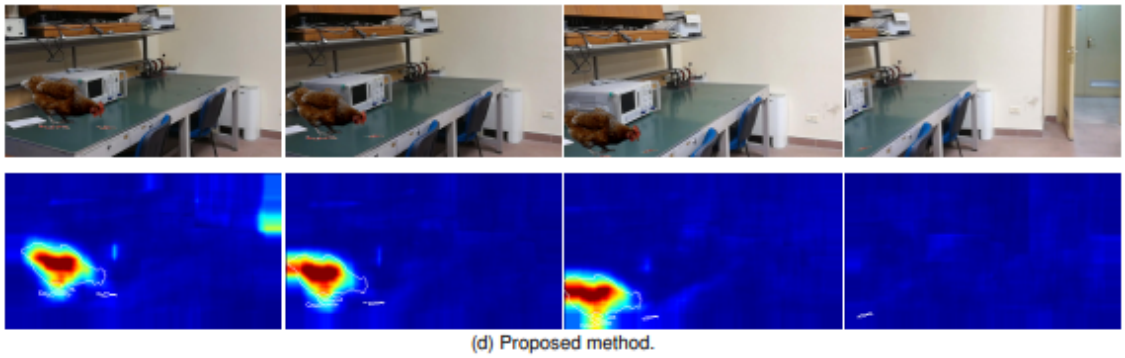


Figura 3.5: Resultados del Estudio: [DCGV17]

3.2.4. Comparación del Estado Actual de la Investigación

En esta sección del capítulo sintetizamos y analizamos los artículos presentados, la Tabla 3.4 resume los resultados, técnicas y datos que utilizaron. Las decisiones de diseño del prototipo, considera los hechos, las hipótesis e ideas descritas en esta sección del trabajo. Estas decisiones y consideraciones se presentan en el Capítulo 4.

Las diferentes técnicas y metodologías que se emplean para la detección de CMF son muy diversas y dispares, no obstante muchos de los estudios comparten cierta similitud a la hora de analizar las imágenes. Las imágenes y/o fotogramas se fragmentan en bloques no superpuestos aunque en algunos estudios esta extracción se realiza empleando una ventana deslizante, en la que los distintos bloques se obtienen desplazando el marco de la ventana, del mismo tamaño que el bloque, píxel por píxel. Todas las lecturas referentes a la detección de CMF que emplean algoritmos de ML fragmentan las imágenes en bloques. Estos bloques son posteriormente procesados y se realiza una extracción de características.

Tabla 3.4: Tabla Comparativa de los Artículos Presentados

Artículo	% accuracy	Estudia	IA	Técnica	Tipo de red	Dataset
[HMS+12]	91.14 %	Imagenes	No	WLD	-	Vídeos propios
[LH13]	99.7 %	Vídeos	No	SSIM	-	Youtube
[RJ16]	98.04 %	Imagenes	Si	SRM	Convolucional	CASIA
[ZGLWT16]	87.51 %	Imagenes	Si	DWT	Autoencoder	CASIA
[DCGV17]	99 %	Vídeos	Si	Detector de anomalías	Autoencoder	UNINA

Existen múltiples técnicas que se aplican para la extracción de características, si observamos nuevamente la tabla 3.4, la precisión media de los estudios presentados es del 95.07 %, siendo la más baja del 87.51 %. Los estudios [HMS+12] y [ZGLWT16] presentan las tasas de precisión más bajas, en las metodologías propuestas, se cambia el espacio de color de las imágenes a procesar de RGB a YCrCb antes de realizar la extracción de las características. La razón por la que el estudio [HMS+12] obtuvo mejores resultados que el estudio [ZGLWT16] puede deberse al descarte de la componente de luminancia (Y) ya que según el propio artículo "la imagen manipulada parece natural, pero pueden quedar rastros manipulados en los canales de crominancia pues el ojo humano es menos sensible a la crominancia que a la luminancia".

El estudio [RJ16] mejoró su precisión un 10.13 % y aceleró la convergencia del modelo inicializando los pesos de la primera capa de la red neuronal. El peso de cada característica se corresponde con un valor que contiene el SRM que emplearon. El mejor resultado le corresponde al artículo [LH13] en el que la detección de manipulaciones se determina gracias a una función que mide la similitud de los fotogramas de una secuencia del vídeo y al cálculo de un valor de umbral dinámico preciso.

De los *datasets* empleados, todos los estudios que han aplicado algoritmos de ML han obtenido sus casos de prueba y validación de los conjuntos de datos proporcionados por universidades y centros de investigación referentes en el campo del estudio forense.

Capítulo 4

Descripción de la Técnica Propuesta

Tras analizar y estudiar las distintas técnicas empleadas en la actualidad, presentamos el algoritmo propuesto. En este capítulo se presentan y fundamentan las decisiones tomadas para la elaboración del programa. A lo largo del desarrollo de este trabajo se han realizado tres implementaciones, siendo estas un evolutivo y mejora de la anterior. Introduciremos primeramente el diseño y conceptualización del primer diseño propuesto y las modificaciones y mejoras realizadas sobre el primer y segundo modelo hasta obtener el algoritmo actual. En el capítulo 5, junto con los experimentos realizados, se presentan los datos de las fases de experimentación iniciales que motivaron estos re-diseños del algoritmo.

4.1. Conceptualización del Algoritmo

Partiendo de los *datasets* proporcionados, analizamos primeramente qué elementos compondrían un caso de prueba. Los *datasets* de imágenes se componen de un conjunto de ternas de imágenes en las que encontramos: la imagen real, la imagen manipulada con CMF y el GT. De forma análoga, los *datasets* de vídeos cuentan con un vídeo original, un vídeo manipulado con CMF y un vídeo que corresponde con el GT. En este GT la zona manipulada es resaltada en blanco.

Con el objetivo de simplificar la implementación, aplicando la lógica del mínimo común múltiplo, tomaremos como caso de entrenamiento una terna de fotogramas. Suponemos que un caso de entrenamiento o evaluación siempre estará compuesto por los siguientes fotogramas (todos ellos en el espacio de color RGB): una *Imagen Original* (IMO), una ***Imagen Manipulada*** **(IMM)** y una imagen de GT que resalte en blanco la zona manipulada sobre un fondo negro. Con las suposiciones anteriores, los casos de prueba

tendrán el aspecto que se ve en la Figura 4.1.

Figura 4.1: Casos de prueba - Ejemplo

Gracias al GT podemos identificar a priori qué zona de la imagen ha sido manipulada y centrar la atención en analizar dicha sección. Puesto que conocemos los resultados (la manipulación) del casos de prueba, implementaremos un algoritmo de AI de entrenamiento supervisado. El resultado de la predicción será una imagen en blanco y negro semejante a la imagen de GT que mostrará la región que ha sido manipulada.

Para ello emplearemos un modelo mixto de red neuronal, en el que mezclaremos la estructura convencional del DAE al que agregaremos una última capa con una única salida para que funcione a modo de clasificador binario. El clasificador binario que retornará un valor entre uno y cero (blanco / negro). Su predicción nos permitirá determinar si un bloque está no manipulado.

El algoritmo tomará como entrada un caso de prueba y devolverá una imagen de GT que se corresponderá con la predicción realizada por la red neuronal. Con la conceptualización anterior se procedió a la implementación del algoritmo y posteriormente a su evaluación.

4.2. Modelo Basado en el Análisis de Imágenes Completas

Este proceso se inicia con la lectura de las imágenes originales y manipuladas (IMO y IMM) que se utilizarán en el proceso de entrenamiento. En el proceso de predicción además de las imágenes (IMO o IMM) que se desean predecir, se utiliza el GT por motivos de depuración y extracción de métricas.

Una vez leídos la imágenes de entrada, se aplica un cambio al espacio de color YCrCb de las IMO y IMM ya que “este espacio de color más sensible a las manipulaciones” [WDT09]. La Figura 4.2 ilustra el cambio de color de una imagen RGB a YCrCb.

Figura 4.2: Cambio de RGB a YCrCb

Los tres fotogramas son posteriormente descompuestos en bloques no superpuestos, obteniendo un listado de ternas de bloques de los distintos fotogramas. Los fotogramas IMO y IMM eran posteriormente descompuestos en las componentes de *Luminance* (Y), *Chroma-Blue* (Cr) y *Chroma-Red* (Cb) a las que se aplicaba la DWT de nivel 3 y se procedía posteriormente a recoger características (la desviación estándar, la media y la suma total) de las descomposiciones horizontales, verticales y diagonales extraídas con la DWT. Todas las características extraídas de los distintos bloques para un fotograma determinado son empleados como vector de entrada del DAE.

Posteriormente, los vectores de características de las imágenes originales y manipuladas eran entrenadas por el DAE. El entrenamiento toma como entrada el vector de características de los distintos bloques del fotograma y trata de asemejar su salida con la imagen de GT proporcionada. En el caso del vector de características de la imagen original la salida trata de asemejarse a una imagen completamente negra.

Finalmente, en el proceso de predicción, se utilizan las imágenes IMO y IMM. La imagen resultante de la predicción obtenida en la red neuronal era posteriormente almacenada en un directorio.

4.2.1. Deficiencias del Modelo

Tras la fase de experimentación inicial del modelo, presentada en el capítulo 5, se identificaron los siguientes defectos:

1. El proceso de extracción de características consume altísimas cantidades de memoria

y los tiempos de ejecución suelen ser elevados.

2. La construcción del DAE está ligado al tamaño del bloque, lo que ocasiona que el algoritmo sea inviable para tamaños de bloque inferiores a 128x128.

3. Problemas de *Overfitting*, los casos de entrenamiento son perfectamente predichos pero los casos nuevos no son coherentes ni fiables.

4. Que la entrada del DAE esté ligada al tamaño del bloque impide que los modelos entrenados sean incompatibles entre procesamientos con distinto tamaño de bloque o imagen.

5. Para probar un set de datos es necesario la elaboración de un script que haga tantas invocaciones del programa como casos de entrenamiento o evaluación queramos hacer.

Esto ocasionaba tiempos de ejecución y consumos de memoria muy elevados, ya que la construcción del DAE estaba ligada al tamaño de los fotogramas y al tamaño del bloque. El programa generaba modelos especializados para una resolución y características determinadas pero eran incompatibles entre vídeos de distinta resolución y/o tamaño de bloque.

4.3. Modelo Basado en el Análisis y Filtrado de Bloques de la Imagen

Para mejorar el modelo anterior se realizaron cambios en los procesos de extracción de características, entrenamiento y predicción.

4.3.1. Extracción de Características

La recepción de los ficheros esperaba recibir una ruta que apuntase a un directorio, el cual debe de contener los vídeos e imágenes que queremos entrenar o predecir.

La construcción del DAE se cambió, el DAE recibiría como entrada los vectores de características correspondientes a un único bloque y retornaría un único resultado que determinaría si el bloque ha sido identificado como manipulado o no. El número de características de un bloque está ahora únicamente relacionado con el nivel con el que aplicamos la DWT. El tamaño de entrada de la red neuronal se rige por la siguiente Ecuación 4.1:

$$f(x) = 9 * (3 * x + 1) \tag{4.1}$$

Siendo x el nivel con el que se aplica la DWT.

Gracias a este cambio, tanto la construcción del DAE como el algoritmo de extracción de características se optimizó, se dio soporte para que el recortado por bloques no superpuestos pudiese emplearse a modo de ventana deslizante. Con cada extracción de un bloque, se extraen inmediatamente sus características y se envía al DAE para almacenar el resultado de la predicción. Pudiendo así, optimizar el problema de memoria del diseño anterior.

4.3.2. Entrenamiento

No todos los bloques tienen el mismo valor a la hora de ser entrenados, a priori un bloque que esté completamente blanco o negro en un fotograma de GT no determina de forma univoca que el bloque haya sido o no manipulado. En el caso de los bloques completamente negros supondremos que estos bloques no han sido manipulados y solamente añaden ruido al entrenamiento. Mientras que los bloques completamente blancos, solamente nos dan indicios de que el bloque está presente en otra región de la imagen, pero el bloque en si no ha sido modificado, debemos de analizar el perímetro de estas zonas que es donde los manipuladores han dejado la huella.

Este filtro basado en el análisis de la composición del bloque de GT nos permite filtrar ternas de bloques que no contienen la información que nos interesa que aprenda la red neuronal, asegurándonos también de que el número de entrenamientos a cero y uno es equivalente. La Figuras 4.3 y 4.4 ilustran en gris los bloques que añaden ruido al entrenamiento.

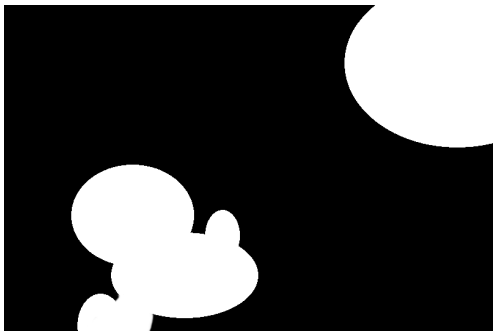


Figura 4.3: Bloques

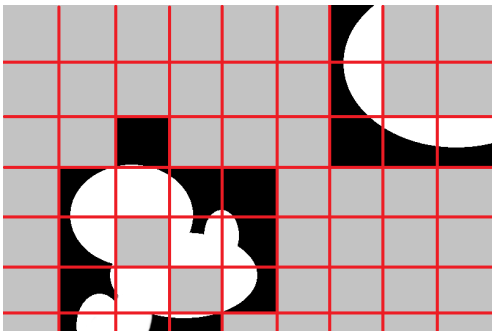


Figura 4.4: Bloques filtrados

El algoritmo de entrenamiento trata de ajustar las características extraídas del bloque IMO con un resultado 0 (negro) y las características extraídas del bloque IMM con un resultado de 1 (blanco).

4.3.3. Predicción

Los resultados de la predicción para las imágenes IMO y IMM son almacenados en una lista. Para cada uno de estos fotogramas, el programa genera una imagen en blanco y negro. Partiendo de una imagen en negro, el programa toma la predicción para cada bloque de la imagen y calcula a partir de estas, un valor R que corresponde con la media de las predicciones. Este valor se emplea para determinar si el bloque es o no coloreado en blanco. Si el valor de la predicción del bloque supera el umbral R el bloque se identifica como manipulado y se pinta de blanco.

En los resultados de los experimentos presentados en la Sección 5.3 se observa que la diferencia entre los valores de un bloque manipulado y no manipulado es muy pequeño ocasionando fallos en la predicción. Para dar solución a este problema se eliminó la capa de clasificación agregada al DAE. Las características finales del DAE son:

- Cinco capas de neuronas
- El tamaño de la entrada y la salida se rige por la Ecuación 4.1 mientras que las capas intermedias se rigen por las Ecuaciones 4.2, 4.3 y 4.4 respectivamente.

$$h1(x) = f(x)/2 \tag{4.2}$$

$$h2(x) = f(x)/6 \tag{4.3}$$

$$h3(x) = h1(x) \tag{4.4}$$

Siendo x el nivel con el que se aplica la DWT y *f(x)* la Ecuación 4.1.

4.4. Descripción del Algoritmo

El algoritmo se compone de dos fases, Una fase de entrenamiento y una fase de predicción. Ambas fases reciben como entrada un directorio con las siguientes especificaciones:

1. El número de ficheros que contiene debe ser múltiplo de tres.
2. Los ficheros deben ser imágenes o en su defecto, vídeos.
3. Los ficheros que conforman los casos de prueba deben ser adyacentes en el listado de ficheros.
4. El nombre del fichero de GT debe contener la cadena "_alpha".

5. El nombre del fichero IMM debe contener la cadena "_copy"

6. Si el directorio contiene vídeos, solo puede contener un único caso de prueba.

Ambos programas siguen el mismo protocolo a la hora de leer el conjunto de fotogramas que contienen los directorios. Una vez identificado un caso de prueba, el algoritmo procede a recorrer el fotograma de GT, horizontal y verticalmente, una cantidad de píxeles Sx e Sy respectivamente, extrayendo bloques de tamaño BxB. Los parámetros S y B están configurados por defecto a 8 y 32 píxeles respectivamente.

En el Algoritmo 1 se presenta la lógica que emplea el algoritmo de ventana deslizante del programa. Nótese la presencia de las funciones **has_to_be_trained** y **process_block**. La función *has_to_be_trained* espera recibir por parámetro un bloque de GT y retorna *True* o *False* si se considera que el bloque ha de ser entrenado, esta condición solamente existe en el programa del entrenador (Algoritmo 2), pues el predictor tiene realizar una predicción de todos los bloques. El comportamiento de la función *process_block* depende del programa; el *Entrenador* en este paso del programa realizará el entrenamiento del modelo, mientras que el *Predictor* construirá el resultado predicho.

Algoritmo 1: Ventana deslizante

```
procedure ProcesarCasoDePrueba(testCase)
    height ← testCase.imgGT.height;
    width ← testCase.imgGT.width;
    h_blocks ← 1;
    w_blocks ← 1;
    if (height − B) > 0 then
        h_blocks += ((height − B)/SX);
    if (width − B) > 0 then
        w_blocks += ((width − B)/SY);
    cols ← 0;
    foreach c in range(h_blocks) do
        rows ← 0;
        foreach R in range(w_blocks) do
            gt_block ← testCase.imgGT[cols:cols + B, rows:rows + B];
            if has_to_be_trained(gt_block) then
                imo_block ← testCase.imgIMO[cols:cols + B, rows:rows + B];
                imm_block ← testCase.imgIMM[cols:cols + B, rows:rows + B];
                process_block(gt_block, imo_block, imm_block);
            rows += SX;
        cols += SY;
end procedure
```

Algoritmo 2: Filtro de bloques

```
procedure DebeSerEntrenado(gt_block)
    color ← cv2.sumElems(gt_block);
    color /= (B * B);
    return color > THRESHOLD_INF && color < THRESHOLD_SUP;
end procedure
```

A la hora de procesar el bloque se procede a extraer las características del bloque actual. Estas características serán posteriormente empleadas por el DAE para el entrenamiento del modelo o la predicción de resultados. Como ilustra el Algoritmo 3, el método de extracción de características espera recibir como parámetro un bloque IMO o IMM del cual se extraen sus componentes y sobre las cuales se aplica la DWT de nivel tres. Posteriormente, se extraen las distintas componentes horizontales, verticales y diagonales resultantes de aplicar la DWT en cada componente YCrCb del bloque. Obtenemos así, un total de 90 características, el número de características viene determinado por la Ecuación 4.1.

Algoritmo 3: Extracción de características

```
procedure ExtraerCaracterísticas(block)
    data ← [ ];
    [y, cr, cb] ← cv2.split(block);
    data += ExtraerComponentes(data, y);
    data += ExtraerComponentes(data, cr);
    data += ExtraerComponentes(data, cb);
    return data;
end procedure

procedure ExtraerComponentes(data, component)
    coeffs ← wavelet(component, wavelet='db2', level=WAVELET_LEVEL);
    data += ExtraerBloque(component);
      foreach i in range(WAVELET_LEVEL) do
      data += ExtraerBloque(coeffs[ i ][ 'horizontal' ]);
      data += ExtraerBloque(coeffs[ i ][ 'vertical' ]);
      data += ExtraerBloque(coeffs[ i ][ 'diagonal' ]);
end procedure

procedure ExtraerBloque(block)
    mean, std ← cv2.meanStdDev(block);
    return [ mean, std, cv2.sumElems(block) ];
end procedure
```

Este conjunto de características representan el vector de entrada del DAE. El DAE cuenta con cinco capas de neuronas, es simétrica y los tamaños de sus capas vienen determinados por las Ecuaciones 4.1, 4.2, 4.3 y 4.4 presentadas anteriormente.

En el programa de entrenamiento, estas características son empleadas para entrenar el modelo, por cada terna de bloques extraída del casos de prueba, el DAE realiza dos entrenamientos; en los que trata de ajustar el bloque de la IMO con una predicción de GT 100 % negra y ajustar el bloque de la IMM con el bloque de GT actual, como ilustra el Algoritmo 4. El programa de predicción recopila las predicciones de la IMO y la IMM. El fotograma de GT se emplea para recopilar estadísticas y calcular la *Matriz De Confusión* (MC) como esquematiza el Algoritmo 5.

Algoritmo 4: Entrenamiento de bloques

```
procedure EntrenaBloque(gt_block, imo_block, imm_block)
    imo_coeffs ← ExtraerCaracterísticas(imo_block);
    imm_coeffs ← ExtraerCaracterísticas(imm_block);
    imo_gt ← zeros(n_outputs);

    DAE.train(feed=imo_coeffs, expect= imo_gt);
    DAE.train(feed=imm_coeffs, expect= gt_block);
end procedure
```

Algoritmo 5: Pintado del resultado

```
procedure PintaResultado(gt_values, predictions)
    result ← zeros(height, width);
    cols ← 0;
      foreach c in range(h_blocks) do
    rows ← 0;
      foreach r in range(w_blocks) do
          if predictions[c * w_blocks + r] > mean_table(predictions) then
              if gt_values[c * w_blocks + r] == 1 then
                  tp += 1;
              else
                  fp +=1;
            result[cols: cols + B, rows:rows + B] = ones(B, B);
          else
              if gt_values[c * w_blocks + r] == 0 then
                  tn += 1;
              else
                  fn += 1;
        rows += SX;
    cols += SY;
    return result;
end procedure
```

4.5. Tecnología Utilizada

El proceso de desarrollo se inició en el entorno de desarrollo *PyCharm* en un computador con las siguientes características:

```
Modelo del ordenador: DELL INSPIRON 5000
Sistema operativo: Windows 10 Home
CPU: i5-4210 @ 1.70 GHz
RAM: 8,00 GB
```

El entorno de desarrollo *PyCharm* favoreció el proceso de aprendizaje sobre el lenguaje *Python*, el manejo de librerías y dependencias así como la depuración y optimización del código desarrollado. La herramienta desarrollada hace uso de las librerías *OpenCV*, *Tensorflow* y *PyWavelets* para la lectura y almacenamiento de ficheros multimedia, la construcción de redes neuronales y la extracción de características DWT respectivamente.

Adicionalmente, la herramienta hace uso de la librería *mathplot* que se emplea para la recogida de estadísticas y depuración de la herramienta.

En una etapa posterior la herramienta se migró a la plataforma *Google Colab*, la cual nos ofrece servidores y máquinas donde alojar y ejecutar los numerosos experimentos que se han llevado a cabo a lo largo de este trabajo. Esta migración estuvo motivada por el tiempo de ejecución de los experimentos realizados en la primera fase de experimentación. Esta plataforma nos ha proporcionado un entorno sencillo y potente desde el que poder evaluar la eficacia del desarrollo sin importar las características del computador.

Capítulo 5

Experimentos y Resultados

En este capítulo se describen los experimentos realizados para evaluar la eficacia del algoritmo propuesto y los resultados obtenidos.

5.1. Composición de los Datasets

Los datasets proporcionados por el equipo de GASS están compuestos por imágenes y vídeos manipulados con CMF. Los vídeos se pueden encontrar en formato MP4 o bien descompuestos por fotogramas. Los datasets proporcionados contienen un total de veinticuatro vídeos y ciento cuarenta y una imágenes. Algunos de los datasets de vídeos presentaban manipulaciones adicionales, como pueden ser giros y deformaciones de la región manipulada.

Adicionalmente, elaboramos cinco vídeos adicionales empleando Adobe Photoshop CS6. Estos vídeos fueron grabados con dispositivos móviles y se manipularon con CMF. Estos vídeos adicionales nos servirían de apoyo para incrementar el número de datos con los que podemos entrenar el DAE y darnos más opciones a la hora de evaluar los modelos.

5.2. Evaluación del Modelo Basado en el Análisis de Imágenes Completas

Este conjunto de experimentos tuvo como objetivo evaluar la eficacia del el modelo inicial propuesto. Los experimentos se realizaron en un ordenador portátil con las siguientes características:

```
Modelo del ordenador: DELL INSPIRON 5000
Sistema operativo: Windows 10 Home
CPU: i5-4210 @ 1.70 GHz
RAM: 8,00 GB
```

Se entrenaron 10 datasets de imágenes, 9 de los cuales se emplearon para el entrenamiento. Fueron entrenados un total de 279 casos de prueba. Los experimentos se realizaron para tamaños de bloque superiores a 32 píxeles. Al ser la entrada de la red neuronal directamente proporcional al tamaño de la imagen y al tamaño del bloque, no se pudieron obtener datos suficientes en las pruebas realizadas para tamaños de bloque inferiores a 32 como para poder contrastar la información y extraer conclusiones de forma objetiva.

Una vez entrenado el modelo se procedió a evaluar los distintos casos de prueba que contenía el dataset restante. Los dataset empleados contenían imágenes manipuladas con CMF y no presentaban otro tipo de transformaciones (rotaciones, deformaciones, ...). Fueron evaluados un total de 31 casos de prueba. Los resultados que se obtuvieron corresponden con los de la Tabla 5.1.

Tabla 5.1: Resultados de los Experimentos Iniciales 1

Tam. bloque	% TP	% TN	% FP	% FN
64	5.10 %	17.25 %	76.76 %	0.89 %
128	4.69 %	20.99 %	72.99 %	1.30 %
256	4.90 %	5.00 %	84.00 %	6.09 %

Estos resultados sugirieron que el DAE no estaba consiguiendo aprender las características deseadas, con el objetivo de depurar y hallar la causa, se realizó la evaluación del modelo con los propios casos de entrenamiento, los resultados se presentan en la Tabla 5.2.

Tabla 5.2: Resultados de los Experimentos Iniciales 2

Tam. bloque	% TP	% TN	% FP	% FN
64	2.89 %	92.61 %	2.13 %	1.83 %
128	3.16 %	93.30 %	1.67 %	0.94 %
256	3.67 %	94.60 %	1.14 %	0.18 %

Contrastando los resultados obtenidos Tablas 5.1 y 5.2, observamos que la red neuronal no aprende a extraer las características deseadas y memoriza los casos de entrenamiento. Este experimento nos permitió confirmar que los casos de prueba y metodología de

entrenamiento nos estaba produciendo un problema de **overfitting** en la predicción de resultados.

Para medir la eficacia de los modelos, consideraremos las medidas de *accuracy* y *precision*, estas medidas se obtienen aplicando las Ecuaciones 5.1 y 5.2, respectivamente.

$$accuracy = \frac{N_{\text{tp}} + N_{\text{fn}}}{N_{\text{tp}} + N_{\text{tn}} + N_{\text{fp}} + N_{\text{fn}}} \tag{5.1}$$

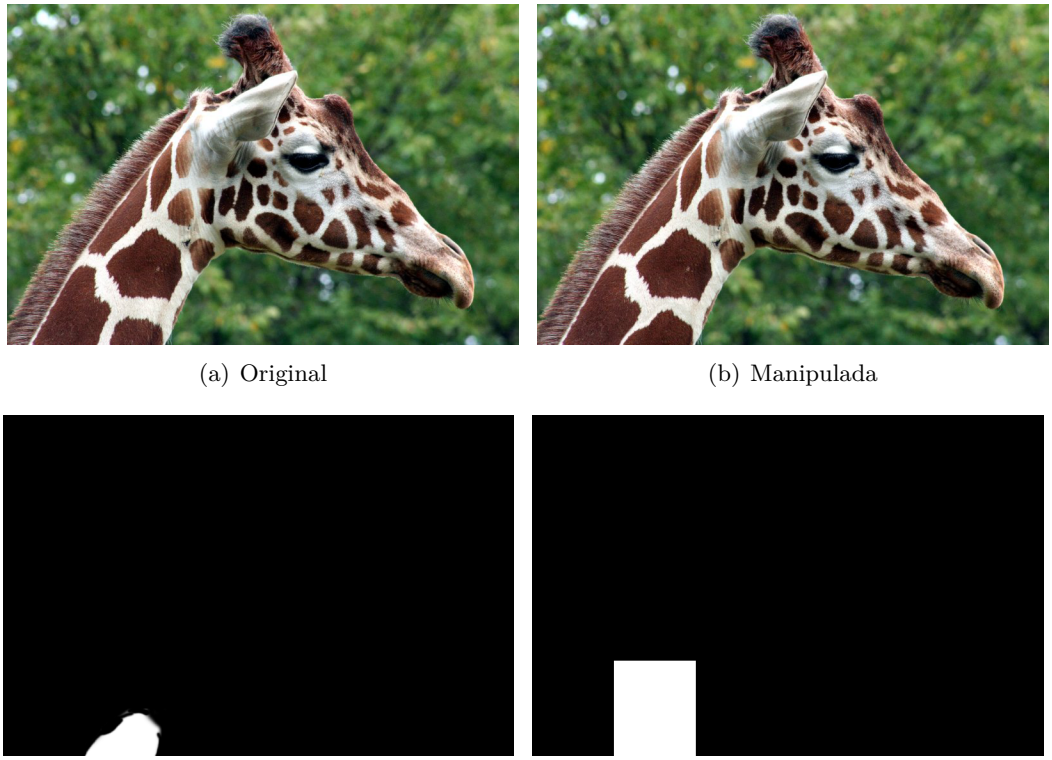
$$precision = \frac{N_{\text{tp}}}{N_{\text{tp}} + N_{\text{fp}}} \tag{5.2}$$

Una vez definidas las medidas, que aplicaremos para determinar la eficacia de los modelos, calculamos la eficacia del algoritmo, obteniendo los resultados que se reflejan en la Tabla 5.3. Ilustramos a continuación dos ejemplos; la Figura 5.1 corresponde con un caso de prueba que no participó en el entrenamiento, la Figura 5.2 ilustra el caso contrario. La gran disparidad de los datos obtenidos en la tasa de aprendizaje y precisión presentados en la Tabla 5.3, junto con las Figuras 5.1 y 5.2 evidencian la incapacidad del programa para predecir correctamente los resultados esperados. Por ello el programa y la política de entrenamiento fue rediseñado.

(a) Original (b) Manipulada

(c) GT (d) Predicción

Figura 5.1: Prototipo Inicial - Caso de Prueba 1



(a) Original (b) Manipulada

(c) GT (d) Predicción

Figura 5.2: Prototipo Inicial - Caso de Prueba 2

Tabla 5.3: Eficacia del Prototipo 1

Casos de prueba	*accuracy*	*precision*
$\in$ Entrenamiento	0.967	0.66
$\notin$ Entrenamiento	0.192	0.056

5.3. Evaluación del Modelo Basado en el Análisis y Filtrado de Bloques

Una vez implementado el segundo modelo, los experimentos se realizaron en la plataforma *Google Colab*. Este algoritmo, como se ha descrito en el capítulo anterior, incorpora un mecanismo para filtrar los bloques que son entrenados y cuyo objetivo es equilibrar los casos positivos y negativos que son entrenados. Cuando entrenamos fotogramas, el bloque de la IMO se ajusta a un bloque negro, mientras que los bloques de la IMM se ajusta con un bloque blanco.

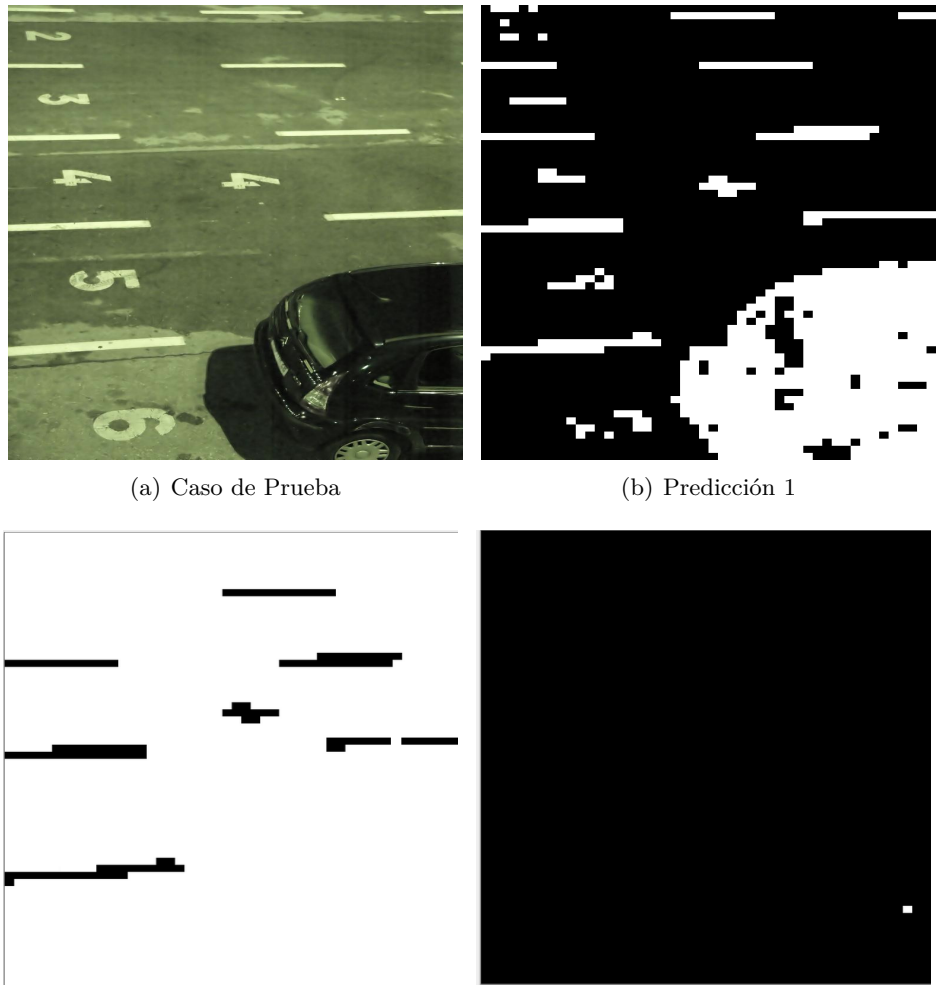
En los experimentos realizados con este modelo se emplearon 10 datasets de imágenes y 5 vídeos. Los primeros experimentos se realizaron con los datasets de imágenes en los

que se emplearon 7 datasets para el entrenamiento y 3 para la evaluación, en una segunda fase, se entrenaron 4 vídeos y se evaluó un quinto.

Ambos datasets estaban compuestos por imágenes y vídeos manipulados con CMF sin ningún tipo de modificación adicional, los resultados de dichos experimentos se muestran en la Tabla 5.4. La Figura 5.3 ilustra diferentes predicciones que se obtuvieron para distintos fotogramas pertenecientes a un mismo vídeo.

Tabla 5.4: Resultados de los Experimentos del Segundo Prototipo 1

Tam. bloque	% TP	% TN	% FP	% FN
8	3.1	65.14	4.26	27.5
16	2.64	49.35	35.01	12.8
32	2.04	51.03	5.86	41.06
64	0.5	56.4	1.3	41.8



(a) Caso de Prueba (b) Predicción 1

(c) Predicción 2 (d) Predicción 3

Figura 5.3: Segundo Prototipo - Caso de Prueba

La Tabla 5.4 cuenta con una tasa de *Negativo* (TN) más elevada que los resultados obtenidos en la Tabla 5.1, no obstante, llama la atención la tasa de *Falso Positivo* (FP) y *Falso Negativo* (FN). Esta oscilación en los resultados sugiere que las predicciones que recibe el algoritmo de pintado no son correctas. Para comprobarlo, se repitieron nuevamente los experimentos pero en este caso, evaluamos los propios casos de entrenamiento para identificar si esta oscilación en la tasa de FP y FN es un problema aislado del algoritmo de pintado o también afecta al DAE. Los resultados de los experimentos se muestran en las Tablas 5.5 y 5.6.

Tabla 5.5: Resultados de los Experimentos del Segundo Prototipo 2

Tam. bloque	**% TP**	**% TN**	**% FP**	**% FN**
8	9.54	75.42	7.03	8
16	9.38	77.03	2.15	11.43
32	8.42	79.04	5.62	9.61
64	7.66	80.4	2.64	9.3

Tabla 5.6: Eficacia del Prototipo 2

Casos evaluados	*accuracy*	*precision*
$\in$ Entrenamiento	0.866	0.677
$\notin$ Entrenamiento	0.572	0.25

Los resultados de la Tabla 5.6 muestran una mejora notable con respecto a los resultados que se obtuvieron con el modelo inicial. La tasa de FP y FN de la Tabla 5.5 es estable, estos datos nos permiten concluir que el problema se encuentra en el modelo.

Para identificar la fuente del problema, se tomaron 97 casos de entrenamiento procedentes de un *dataset* de imágenes y se procedió a contrastar el valor de los bloques del GT (valor ideal) con los valores predichos por la red neuronal. En verde se resalta el valor teórico ideal que debería retornar la predicción para los distintos bloques. En rojo, el valor predicho por la red neuronal. Si observamos la Figura 5.4 podemos observar que las predicciones de la red neuronal son prácticamente idénticas tanto para bloques manipulados como originales. En condiciones ideales la línea roja debería de ajustarse a la gráfica verde, esto nos permitiría discriminar los bloques manipulados con mayor criterio y seguridad.

Estos resultados indican que el DAE no discrimina correctamente los bloque manipulados de los originales. Como la diferencia entre la predicción de un bloque manipulado y uno no manipulado es demasiado pequeña, el algoritmo de pintado que interpreta los resultados del DAE hace uso de un umbral que se calcula como el valor

medio de las predicciones de los bloques para una imagen en particular. Esto ocasiona una gran oscilación de FP y FN en los resultados obtenidos.

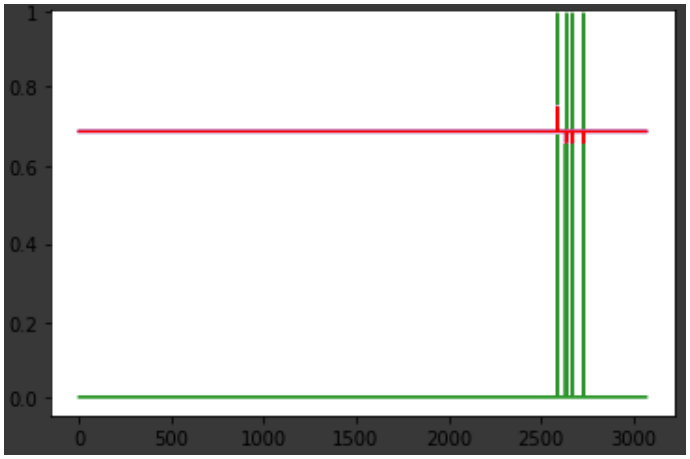


Figura 5.4: Predicciones del DAE - Prototipo 2

5.4. Evaluación del Algoritmo Propuesto

Esta fase de experimentación se ha realizado en la plataforma *Google Colab*. En esta fase de experimentación, se han realizado experimentos con datasets de vídeos e imágenes.

Esta fase de experimentación se ha desarrollado en tres fases principales:

5.4.1. Obtención de los Parámetros de Entrenamiento y Predicción

Los parámetros de ejecución de esta fase de experimentación se han heredado de los experimentos realizados con el segundo modelo, describimos a continuación el proceso realizado para la obtención de dichos parámetros.

Con el objetivo de ejecutar el programa en las condiciones ideales y óptimas, debemos de realizar experimentos que nos permitan determinar qué configuración favorece este tipo de resultados. La herramienta consta de los siguientes parámetros de ejecución:

1. **Umbral de filtro de bloques inferior**: Este parámetro toma valores en el rango [0-255] y especifica la proporción mínima de blanco que debe contener un bloque para ser entrenado.

2. **Umbral de filtro de bloques superior**: Este parámetro toma valores en el rango [0-255] y especifica la proporción máxima de blanco que debe contener un bloque para ser entrenado.

3. **Nivel con el que se aplica la DWT**: Este parámetro, al que nombraremos de ahora en adelante `NIVEL_WAVELET`, toma valores enteros; cuanto mayor sea su valor mayor será el número de descomposiciones que obtengamos de los bloques.

4. **Tamaño de bloque**: El parámetro `TAM_BLOQUE` especifica el tamaño en píxeles del bloque. Tamaños menores de bloque conllevan tiempos de ejecución más elevados.

5. **Desplazamiento del bloque**: El parámetro `DESP_BLOQUE` especifica en píxeles la cantidad de píxeles que se desplaza el bloque cuando el programa fragmenta las imágenes, el valor de este parámetro no debe de ser inferior a uno ni superior al tamaño del bloque.

Se elaboró un experimento inicial que tuvo por objetivo medir los parámetros ideales para los umbrales del filtrador de bloques. Este experimento se realizó entrenando imágenes y visualizando la cantidad de bloques que superaban el filtro. Se busca un resultado en el que los bloques con una gran proporción de negro y blanco sean descartados, al no contener rastros de cortes ni regiones difuminadas ni suavizadas, para evitar el problema de sobreajuste del primer diseño.

El experimento concluyó que valores entre [25-40] y [215-225] son ideales para los umbrales de filtro inferior y superior, respectivamente. Estos valores nos permiten filtrar con seguridad todos aquellos bloques con una proporción de negro y blanco superior al 90 %, obteniendo bloques como los iustrados en la Figura 5.5.

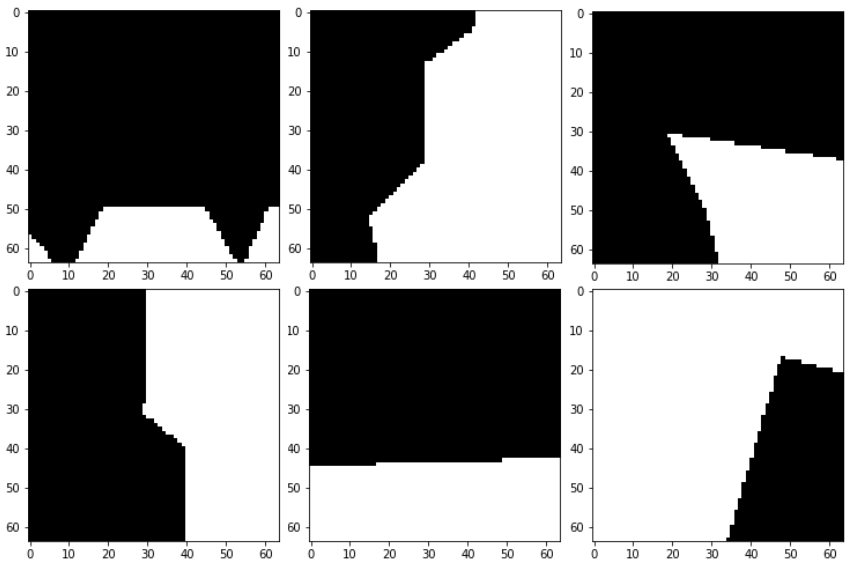


Figura 5.5: Experimento sobre el Filtrado de Bloques

Para la obtención de los parámetros: `TAM_BLOQUE` y `DESP_BLOQUE` se realizaron numerosos experimentos con distinto tamaño de bloque y desplazamiento. Los resultados de este experimento se corresponden con los resultados presentados en la Tabla 5.4. A lo largo de esta fase experimental se observó una mayor tasa de acierto y precisión cuanto menor era el tamaño de bloque, no obstante los experimentos que se realizaron no nos permiten determinar con certeza qué valores son los más óptimos en cuanto a precisión y tasa de aciertos. Solamente podemos afirmar estos valores parecen efectivamente representar un compromiso entre precisión y tiempo de ejecución.

En lo que respecta al parámetro `NIVEL_WAVELET`, se observó un empeoramiento del 6 % en la tasa de aciertos cuando se experimentó con el nivel 2, niveles mayores no parecieron ofrecer mejores resultados.

La configuración final adoptada para la ejecución de los experimentos es la siguiente:

```
Umbral inferior: 35
Umbral superior: 220
NIVEL_WAVELET: 3
TAM_BLOQUE: 32
DESP_BLOQUE en eje X: 8
DESP_BLOQUE en eje Y: 8
```

5.4.2. Análisis de la Media Aritmética como Umbral de Clasificación de Bloques Manipulados

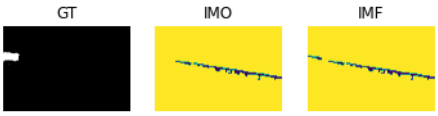
En esta fase de experimentación con imágenes se han entrenado 18 datasets con imágenes manipuladas con CMF. Posteriormente, el modelo entrenado ha sido evaluado con 4 datasets de imágenes. Las imágenes “a”, “b” y “c” de las Figuras: 5.6, 5.7 y 5.8 corresponden la IMO, la IMM y la captura de los resultados proporcionados por el modelo.

(a) Original

(b) Manipulada

```
Info: IMO - TP: 52 TN: 429 FP: 9199 FN: 0
Info: IMO - Accuracy: 4.96900826446280 95%
Info: IMO - Precision: 0.56210139444384 39%
Info: IMO - Recall: 100.0%

Info: IMF - TP: 48 TN: 464 FP: 9164 FN: 4
Info: IMF - Accuracy: 5.2892561983471 08%
Info: IMF - Precision: 0.52105948762483 71%
Info: IMF - Recall: 92.30769230769 23%
```



(c) Predicción

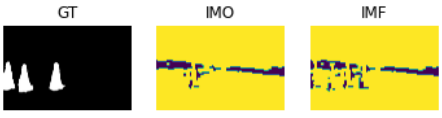
Figura 5.6: Prototipo Final - Caso de Prueba 1

(a) Original

(b) Manipulada

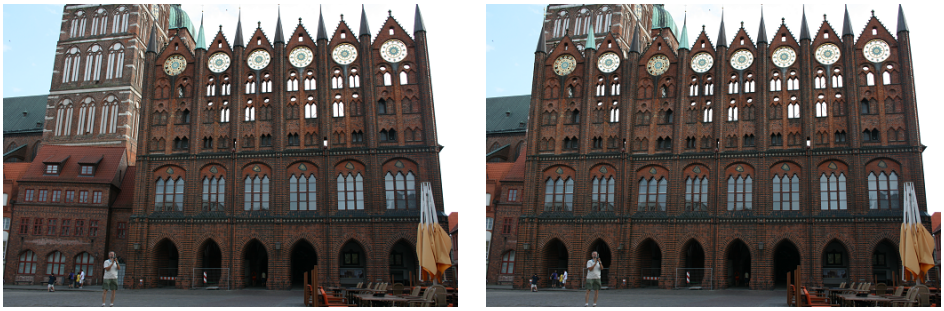
```
Info: IMO - TP: 167 TN: 518 FP: 5095 FN: 48
Info: IMO - Accuracy: 11.7536032944406 32%
Info: IMO - Precision: 3.1736982136069 94%
Info: IMO - Recall: 77.67441860465117%

Info: IMF - TP: 124 TN: 640 FP: 4973 FN: 91
Info: IMF - Accuracy: 13.1091283459162 65%
Info: IMF - Precision: 2.4328036099666472%
Info: IMF - Recall: 57.6744186046511 66%
```



(c) Predicción

Figura 5.7: Prototipo Final - Caso de Prueba 2



(a) Original (b) Manipulada

```
Info: IMO - TP: 125 TN: 4787 FP: 4532 FN: 236
Info: IMO - Accuracy: 50.743801652892564%
Info: IMO - Precision: 2.6841314150740824%
Info: IMO - Recall: 34.626038781163444%

Info: IMF - TP: 140 TN: 4916 FP: 4403 FN: 221
Info: IMF - Accuracy: 52.23140495867768%
Info: IMF - Precision: 3.0816640986132513%
Info: IMF - Recall: 38.78116343490305%
```

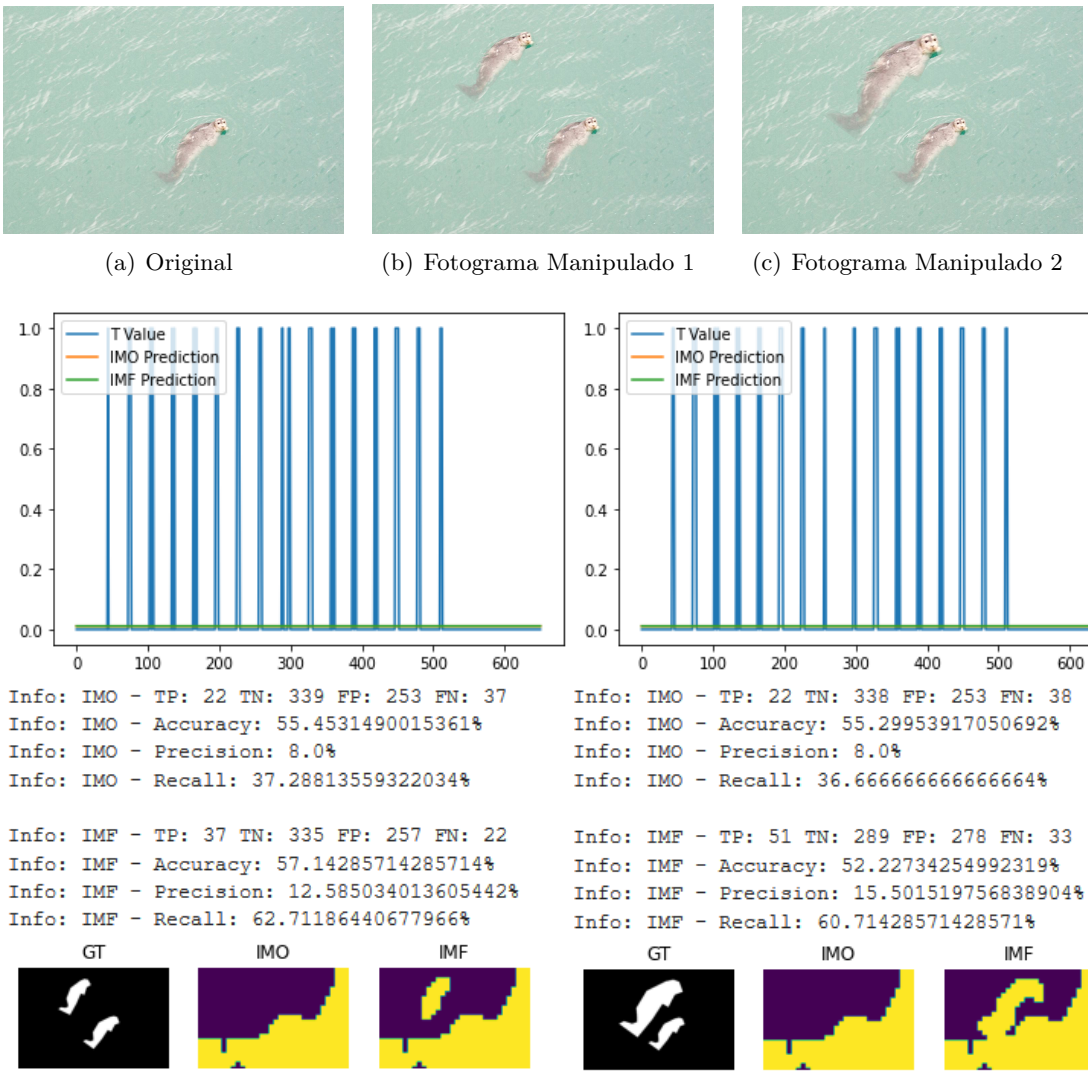
(c) Predicción

Figura 5.8: Prototipo Final - Caso de Prueba 3

Los experimentos con imágenes obtienen resultados con tasas de acierto y precisiones dispares por lo que es difícil determinar con exactitud la precisión real del modelo, sin embargo pese al gran número de FP, las imágenes de GT predichas para la IMO y la IMM suelen desvelar la región manipulada si restamos dichas imágenes. La tasa de aciertos más elevada obtenida en esta fase de experimentación ronda el 54 %.

En la fase de experimentación con vídeos se han llevado a cabo dos experimentos: uno de ellos empleando el modelo entrenado en la fase anterior y otro entrenando un nuevo modelo con 8 datasets de fotogramas manipulados con CMF. Los resultados que se han obtenido en ambos experimentos no presentaban grandes variaciones. La Figura 5.9 muestra los resultados que retornó el modelo durante su evaluación empleando el modelo entrenado en la fase anterior.

Los experimentos con vídeos obtienen tasas de acierto mucho más elevadas que con imágenes. La tasa media registrada más elevada durante la evaluación de un vídeo ha sido del 59,4 %. En la Figura 5.9 se contrastan los resultados obtenidos para dos fotogramas del vídeo. Los experimentos que se han realizado, pese a obtener tasas de FP altas, parecen detectar las manipulaciones con la misma eficacia cuando estas regiones presentan giros o cambios de tamaño.



(a) Original (b) Fotograma Manipulado 1 (c) Fotograma Manipulado 2

(d) Predicción

Figura 5.9: Prototipo Final - Caso de prueba 4

5.4.3. Optimización del Umbral de Clasificación de Bloques Manipulados

Como se observa en las gráficas de la Figura 5.9, el valor para las predicciones de los bloques de la IMO y IMM es "idéntico". En el caso de las predicciones para la IMM estos valores no se corresponden con los esperados (TValue). Para ello se ha realizado un experimento que ha tenido por objetivo: analizar las distribuciones de las predicciones tanto de bloques originales como manipulados y la elaboración de una tabla que permita obtener un valor de umbral en función de la media y el máximo valor de las predicciones.

Este experimento se ha realizado empleando 3 data sets de vídeos y 3 de imágenes para los cuales se han obtenido un total de 588 medias, 6.800.808 predicciones de bloques

originales y manipulados. Para el análisis de estos resultados agrupamos las predicciones obtenidas para bloques manipulados y originales por rangos de media. Las predicciones obtenidas están comprendidas en el rango [0.01042 - 0.0110]. A continuación se sumaron los valores de las predicciones por grupo y tipo de bloque, obteniéndose así dos grupos de valores por media para las IMO y IMM.

La Tabla 5.7 resume los resultados del experimento. El valor de la columna con el nuevo valor del Umbral contiene una estimación más ajustada para las predicciones de los bloques manipulados. En los experimentos también se ha observado que para las predicciones que contienen bloques cuya estimación alcanzan el máximo, el valor de la media no se corresponde con el estimado en la tabla. Para ello, en caso de que alguna predicción de la imagen alcance el máximo, se aplica una corrección a la media ilustrada en la tabla sumándole al umbral correspondiente el valor 0.00003. Este valor se obtuvo estudiando las desviaciones y varianzas de las medias en fotogramas en las que alguno de sus bloques alcanzó el máximo y fotogramas que seguían el comportamiento descrito en la Tabla 5.7.

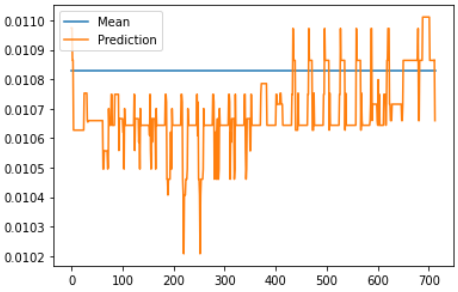
Tabla 5.7: Resultados del Análisis de la Distribución de Predicciones

Media artimética	Nuevo valor de Umbral
> 0.0107	0.01087
> 0.0106	0.01077
> 0.0104	0.01072
else	0.0107

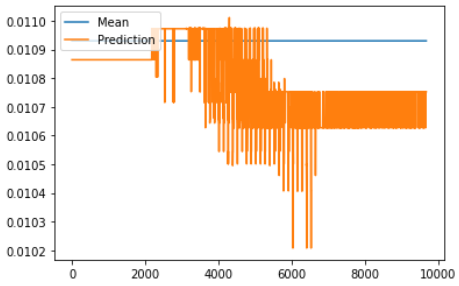
Tras la implementación de esta tabla en el algoritmo de pintado, se procedió a evaluar el nuevo algoritmo de pintado. Los resultados han sido los siguientes:

Las gráficas presentadas en la Figura 5.10 representan el valor de las distintas predicciones para cada bloque en color naranja y el valor de umbral en azul. El nuevo valor de umbral reduce considerablemente el número de bloques que cumplen la condición para ser considerados manipulados, por lo que en los resultados debemos de esperar un número menor de FPs.

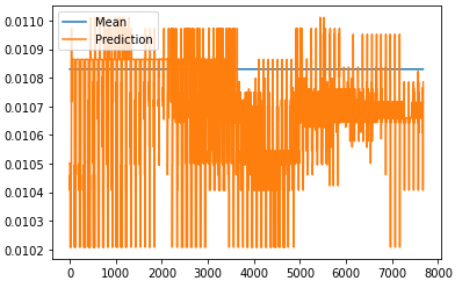
Las Figuras 5.11 y 5.12 son ejemplos de los resultados obtenidos. Estos resultados se han obtenido empleando el mismo modelo entrenado que en el experimento 5.4.2. Esta nueva estimación del umbral supone una mejora considerable para el algoritmo de pintado, si comparamos el resultado obtenido en la Figura 5.6 con los resultados obtenidos en la Figura 5.12 observamos una mejora en la predicción de esta imagen del 76 %. En términos relativos la optimización realizada al umbral de pintado ha mejorado la predicción de imágenes un 32 %, siendo la tasa de aciertos en imágenes del 85,89 %. En vídeos se observa también una mejora del 17 % siendo la tasa de aciertos en vídeos del 76 %.



(a) Gráfica 1



(b) Gráfica 2



(c) Gráfica 3

Figura 5.10: Umbral del Algoritmo de Pintado

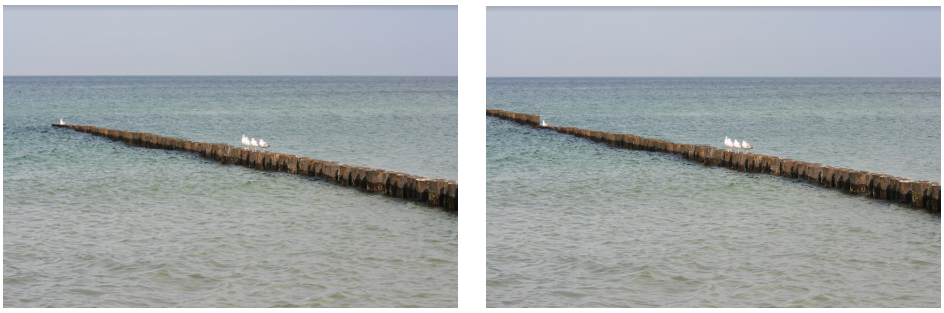
(a) Original

(b) Manipulada

```
Info: MEAN = 0.010585745395135461
Info: IMO - TP: 0 TN: 4125 FP: 1641 FN: 0
Info: IMO - Accuracy: 71.540062434963578
Info: MEAN = 0.01055910431388615
Info: IMF - TP: 28 TN: 4182 FP: 1445 FN: 111
Info: IMF - Accuracy: 73.01422129725988
Info: IMF - Precision: 1.9008825526137134%
Info: IMF - Recall: 20.143884892086338
```

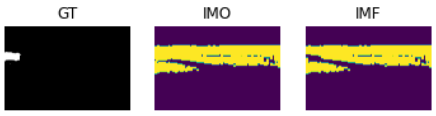
(c) Predicción

Figura 5.11: Resultados del Algoritmo de Pintado Mejorado 1



(a) Original (b) Manipulada

```
Info: MEAN = 0.010795188075718458
Info: IMO - TP: 0 TN: 7692 FP: 1988 FN: 0
Info: IMO - Accuracy: 79.46280991735537%
Info: MEAN = 0.010793013140190991
Info: IMF - TP: 25 TN: 7735 FP: 1893 FN: 27
Info: IMF - Accuracy: 80.16528925619835%
Info: IMF - Precision: 1.3034410844629822%
Info: IMF - Recall: 48.076923076923076%
```



(c) Predicción

Figura 5.12: Resultados del Algoritmo de Pintado Mejorado 2

5.5. Análisis y Contraste con el Estado del Arte

Los resultados obtenidos en esta tercera fase de experimentación presentan mejoras respecto a las del segundo modelo. Eliminar la capa de clasificación con la que diseñamos el DAE nos ha permitido dar solución a problemas como el ilustrado en la Figura 5.3. La optimización del umbral del algoritmo de pintado ha favorecido una mejor interpretación de las predicciones y una mejora de los resultados sustancial.

Tras analizar los resultados hemos concluido que eliminar la capa clasificatoria, aunque no nos haya permitido mejorar notablemente los resultados, ha mejorado de forma significativa las predicciones del modelo. En la Figura 5.9 se ilustran unas gráficas generadas por el algoritmo en la que se muestran: El valor teórico de los bloques, la predicción de la IMO y la predicción de la IMM para cada bloque. La Figura 5.13 se corresponde con la misma gráfica generada por el algoritmo a la que se le ha filtrado la línea de valor teórico de los bloques.

La gran proximidad de las predicciones en el espacio de valores sugiere que el prototipo presenta un problema en el modelo con el que se entrena el DAE. Las características actuales no permiten discriminar con criterio los bloques manipulados de los originales, las predicciones para ambos bloques debe ser distante en el espacio para que el modelo ofrezca mejores resultados.

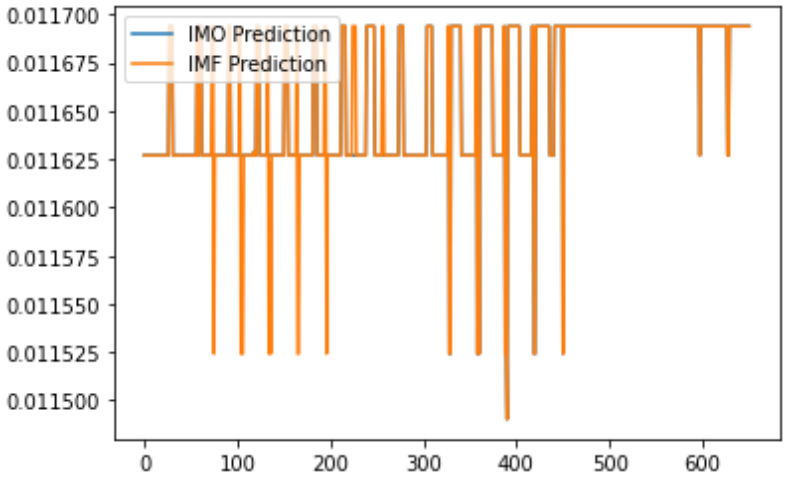


Figura 5.13: Predicciones del DAE - Prototipo Final

Contrastando los resultados del algoritmo propuesto con el Estado del Arte, resumido en la Tabla 3.4, se observa que la tasa de aciertos del algoritmo propuesto alcanza el 86 %, la cual dista de las mencionadas en el Estado del Arte, no obstante debemos de tener en cuenta que los estudios presentados en el Estado del Arte solamente se emplean para la detección en imágenes o vídeos mientras que el programa procesa por igual las imágenes y los vídeos.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Conclusiones

Una vez finalizado el trabajo, extraemos las siguientes conclusiones:

Tras estudiar los distintos artículos recopilados a lo largo de la fase de investigación nos encontramos con una colección de técnicas amplia y heterogénea sobre cómo detectar manipulaciones CMF en imágenes y vídeos. Este primer contacto con el estado del arte nos hizo comprender la complejidad del problema que tratábamos de resolver.

El diseño del algoritmo se ha centrado en elaborar un algoritmo que sea efectivo para detectar las manipulaciones CMF con independencia de que evaluemos imágenes o vídeos. El propósito del algoritmo es tratar de identificar aquellos bloques que son susceptibles de presentar signos de edición, como pueden ser: los recortes y regiones suavizadas. Para ello se optó por emplear las características que se obtenían con la DWT; las cuales nos permiten extraer las primitivas horizontales, verticales y diagonales de los potenciales cortes presentes en los bloques. El desarrollo de la herramienta nos ha permitido aprender y poner en práctica una gran cantidad de conocimientos, especialmente en el campo de la AI, con los que previamente nunca habíamos trabajado.

Durante el desarrollo del trabajo se han elaborado tres modelos. Siendo cada uno de ellos, un evolutivo del anterior. Estos re-diseños fueron motivados no solamente por los resultados que se obtuvieron en las correspondientes fases de experimentación, con tasas de acierto del 19 % y 57 % respectivamente, sino porque fuimos perfeccionando los conocimientos de *Python* y realizamos numerosas optimizaciones que han reducido considerablemente el consumo de memoria y tiempo de ejecución del último algoritmo con respecto a su diseño inicial. Estas optimizaciones posibilitaron la realización de pruebas con mayor rapidez y consecuentemente la obtención de mayores cantidades de datos con los que contrastar ejecuciones y extraer conclusiones.

Los numerosos experimentos realizados a lo largo de los diferentes modelos sugieren que los resultados del algoritmo no parecen verse afectados por los deformaciones o giros que puedan presentar las regiones manipuladas. No obstante, el algoritmo parece ser menos efectivo con fotogramas comprimidos para ejecuciones con un tamaño de bloque reducido. La configuración de los umbrales inferiores y superiores del filtro de bloques del programa entrenador permite filtrar bloques que no contengan el porcentaje de región manipulada deseado. Los experimentos han dado mejores tasas de resultados para configuraciones de umbrales del 15 % y 85 % para los umbrales inferior y superior respectivamente.

Los experimentos concluyen que las métricas y características extraídas por bloque han demostrado ser significativas. Ofreciendo el último modelo una tasa de acierto del 86 %. La tasa de éxito es ligeramente inferior a la que obtuvieron los trabajos de investigación presentados en el estado del arte, no obstante debemos de tener en cuenta que las técnicas propuestas en estos estudios funcionan exclusivamente con vídeos o imágenes y que el modelo trabaja indistintamente con ambos tipos de fichero. Lo cual marca un punto de partida hacia el desarrollo herramienta unificada que identifique manipulaciones CMF en ficheros de imagen y vídeo.

6.2. Trabajo Futuro

Para mejorar la precisión del modelo debemos de poder identificar con mayor claridad en el espacio de valores qué predicciones corresponden con bloques IMO y bloques IMM, para ello se propone como posibles trabajos futuros:

- Agregar al modelo las características resultantes de aplicar la DCT a los bloques y características de los bloques adyacentes (características DWT y DCT). Deberíamos evaluar si estas características adicionales favorecen que el DAE discrimine con mayor criterio los bloques de los fotogramas.

- Eliminar la componente Y del modelo que, acorde con el artículo [HMS+12], es la componente YCrCb a la que los ojos son más sensibles y por lo tanto los rastros de manipulación suelen ser menores, siendo más probable detectar manipulaciones analizando las componentes de crominancia.

- Investigar qué características podemos agregar al modelo que permitan identificar regiones suavizadas o difuminadas, susceptibles de haber sido manipuladas con alguna herramienta de edición.

Resumen del Trabajo en Inglés

Capítulo 7

Introduction

7.1. Motivation

In the last century scientific and technological development has strongly changed our society, making possible the automatization of processes by using computers and electronic devices. The mechanization of industries and administrative tasks is deeply linked to the economic, social and technological boom of our society. This surge has made possible that the most the population in developed countries have access to this technology easily.

Since its origins, the Internet hasn't stopped growing; it was originally built as a computers network to connect scientific communities in the United States, England and France. Nowadays the Internet has become a network of networks, a decentralized network with global scope that offers us a multitude of resources and services, such as: access to news breadcasting platforms through web pages or log on «social networks». We should understand a social network as any kind of service or app which allows us to create our own user profile from which we can upload content; such as images or videos, create a list of contacts, share and view the content that other contacts have uploaded. Some examples are: Facebook, Twitter or Instagram.

Internet has become a part of many people's lives [Hay01]. Access to information is practically immediate thanks to the new technologies; both official news web pages as well as social networks have become an unprecedented way of spreading information, being almost instantaneous and, in most cases, for free.

If we only consider the population that has a mobile device; 7 out of 10 people in the world have mobile phones. Out of these 7 people, approximately 71 % of them have social networks, that is 5 out of 10 people in the world [Kem20].

Videos that have no obvious editing signs or traces are often regarded as part of the

truth as they are real events captured by electronic devices. However, with the development of technology, there are powerful and sophisticated tools that impressively facilitate the editing of digital videos, even for those who do not have technical or specialized knowledge in the filed [GKWB07].

When a large number of edited and falsified videos appear onto social media or broadcast media, there is no doubt that they will have a significant adverse effect on the stability of society. Therefore, the forensic study of digital videos has become an extremely important research topic.

7.2. Context

This Final Degree Project has been carried out within the Analysis, Security and Systems Group (Group GASS, https://gass.ucm.es/, Group 910623 of the catalog of groups recognized by the UCM) as part of the activities of the research project THEIA (Techniques for Integrity and Authentication of Multimedia Files of Mobile Devices) with reference FEI-EU-19-04.

7.3. Research Purpose

When we talk about manipulation in videos, we refer to those which have been manipulated with the purpose of fooling others by altering the reality of the facts they reflect.

The continuous development of technology facilitates the recording of videos as well as their edition, therefore it is essential to develop a tool capable of detecting manipulations. These manipulations leave their mark on the different frames of the video [SM16], in this study we propose an AI algorithm based on the research carried out over the past few years on the detection of CMF in images and videos.

7.4. Work Schedule

The work's development has been carried out in three phases following a circular methodology based on the scientific method:

1. **Investigation**: The aim in this phase was to acquire the necessary technical knowledge on the domain of the problem we are trying to solve. This knowledge would allow us to brainstorm ideas and hypotheses about the algorithm to be developed.

Therefore, meetings were held during the first months of the project in order to find out the background of the problem and the doubts that arose during the research and also with the technology to be used and the development of neural networks.

2. **Development**: In this phase our main objective was to develop the design proposed in the previous phase. Initially the level of knowledge was not enough to tackle the problem, since the subject of Machine Learning was necessary. To learn more about AI before developping the prototype. That is the reason why I participated in conferences and workshops that, along with the information and tutorials that we found on the Internet. All this new knowledge allowed us to develop the different prototypes presented in this work. The developments has been made using the programming language *Python* and with the help of the following libraries: *Tensor Flow* and *OpenCV*. Initially the prototype was developed using *PyCharm* but later it was adapted to run on the *Google Colab* platform.

3. **Experimentation**: The objective in this phase was the evaluation of the designed prototype and the subsequent analysis of the results. These phases of experimentation allowed us to detect development errors and improve the implementation of the proposed prototype. The training and evaluation sets were built from the data sets provided by the GASS team.

4. **Documentation**: The documentation of the work has been treated as a different task from the mentioned work schedule. This task began shortly after having read the first articles that were considered a reference to our work, this task initially consisted on writing down phrases and quotes from different articles and books. Since the knowledge about AI and the *Python* language; we prioritized the development of the prototype, the drafting of the work begun after the first phase of experimentation was completed.

In the Table 7.1 we summarize the tasks performed throughout the development of the project. Additionally, a Gantt chart illustrating the tasks and their timing is attached in Figure 7.1.

Table 7.1: Tasks Performed Throughout the Project

Identifier	Task
1	Investigation
1.1	Reading and Research of academic articles
1.2	Solution design
1.3	Study and analysis of overfitting
2	Development
2.1	Prototype development in Google Colab
2.2	Optimization of the prototype
3	Experimentation
3.1	Extracting components from images and videos
3.2	Image Experiments
3.3	Video Experiments
3.4	Analysis of the results
4	Documentation
4.1	Annotations, citations and article compilation
4.2	Drafting

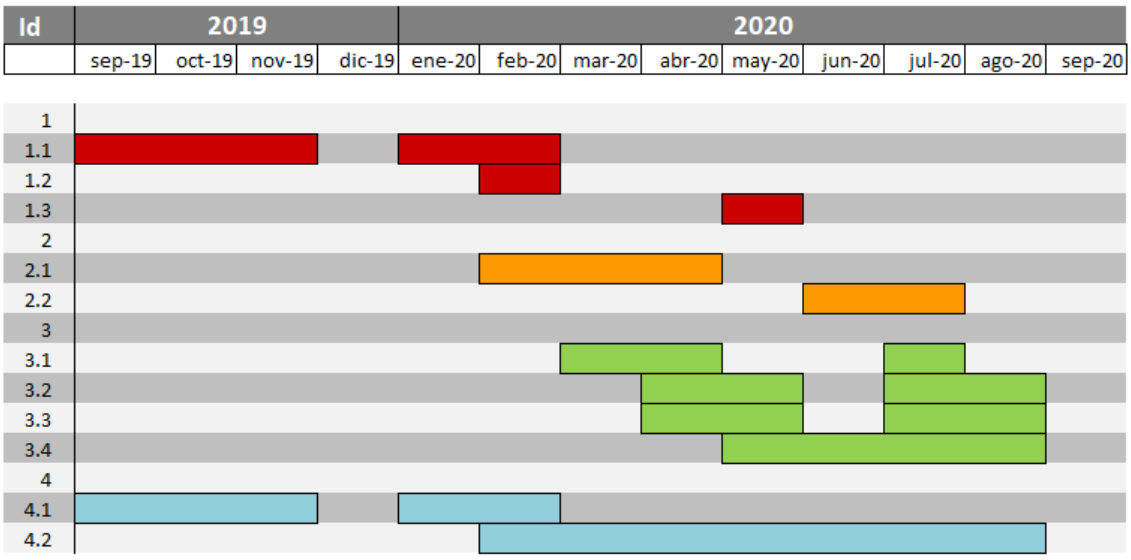


Figura 7.1: Gantt Diagram

7.5. Work Structure

The rest of the work is organized in seven chapters with the structure discussed below:

Chapter 2 contextualizes and briefly introduces the concepts and theoretical bases of the work. Specifically, it describes the functioning of digital cameras and the process of digitizing and recording videos. Next we introduce the basics of AI and finally, we present the different types of manipulations that are related to the CMF manipulation.

Chapter 3 briefly presents the different manipulation detection techniques used in the forensic study of images and videos. Next the state of the art, articles and investigations which have applied different techniques to detect CMF in images and videos.

Chapter 4 presents the AI algorithm proposed of this work as well as the decisions and considerations that were taken in the design process of the prototype.

Chapter 5 describes the experiments performed to evaluate the algorithm's efficiency proposed in Chapter 4. The results are analyzed and subsequently contrasted with those of the academic articles presented in Chapter 3.

Chapter 6 concludes this work by briefly summarizing the project experience and analyzing the results the proposed prototype obtained. Future work and possible research paths are also presented.

Chapters 7 and 8 are the English translation of Chapters 1 and 6.

Capítulo 8

Conclusions and Future Work

8.1. Conclusions

Once the work finished, we draw the following conclusions:

After studying the different articles collected throughout the research phase, we found a wide and heterogeneous collection of techniques on how to detect CMF manipulations in images and videos. This first contact with the state of the art made us understand the complexity of the problem we were trying to solve.

The design of the prototype was focused on developing an algorithm that is effective when it tries to detect CMF manipulations regardless of whether we evaluate images or videos. The purpose of the algorithm is to try to identify those blocks that are susceptible of editing, such as: cuts and smoothed regions. For this, we decided to use the characteristics obtained with the DWT; which allow us to extract the horizontal, vertical and diagonal primitives of the potential cuts present in the blocks. The development of the tool has allowed us to learn and put into practice a large amount of knowledge, especially in the field of AI, with which we had never previously worked.

During the development of the work, three prototypes have been produced. Each one was an evolutionary of the previous one. These re-designs were motivated not only by the results obtained in the corresponding experimentation phases, 19 % and 57 % success rate respectively, but also because we improved our knowledge of *Python* and carried out numerous optimizations which have considerably reduced the memory consumption and runtime of the prototype compared to its initial design. These optimizations made possible to carry out tests more quickly and consequently obtain greater amounts of data with which the working team cloud contrast executions and draw conclusions.

The numerous experiments carried out throughout the different prototypes suggest

that the results of the prototype do not seem to be affected by the deformations or twists that the manipulated regions may present. However, the algorithm appears to be less effective with compressed files for low-block-size runs. The configuration of the lower and upper thresholds for the block filter of the trainer program allows us to filter blocks that do not contain the desired percentage manipulated region. Experiments have yielded better result rates for threshold settings of 15 % and 85 % for the lower and upper thresholds respectively.

Our experiments conclude that the metrics and characteristics extracted per block have proven to be significant. Our last model has given a success rate of 86 %. The success rate is slightly lower than the researches that were introduced in the state of the art, but we have to bear in mind that the studies presented works exclusively with videos or images and that our prototype works indifferently with both of them. The algorithm proposed marks a starting point towards the development of a unified tool that detects CMF manipulations in images and videos files.

8.2. Future Work

To improve the precision of the model, we must be able to identify more clearly which predictions belongs to IMO blocks and which ones to IMM blocks. It is proposed as possible future works:

- Add to the model the characteristics resulting from applying the DCT to the blocks and characteristics of the adjacent blocks (characteristics DWT and DCT). We should evaluate if these additional characteristics favor the DAE to discriminate with more criteria the blocks of the frames.
- Remove the Y component from the model who, according to the article [HMS+12], it is the YCrCb component to which our eyes are most sensitive and therefore the traces of manipulation are usually lower, being more likely to detect manipulations analyzing the chrominance components.
- Research what characteristics can be added to the model in order to improve detection over blurred or smoothed regions which may have been forged via editing tools.

Bibliografía

[Coz16] L. Cozzolino, D.and Verdoliva. Single-image splicing localization through autoencoder-based anomaly detection. In *Proceedings of the IEEE International Workshop on Information Forensics and Security*, pages 1–6, 2016.

[CSH+09] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikäinen, X. Chen, and W. Gao. WLD: A Robust Local Image Descriptor. In *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence*, Agosto 2009.

[DCGV17] D. D'Avino, D. Cozzolino, P. Giovanni, and L. Verdoliva. Autoencoder with Recurrent Neural Networks for Video Forgery Detection. In *Proceedings of the Electronic Imaging*, 2017.

[FSJ03] J. Fridrich, D. Soukal, and L. Jan. Detection of copy-move forgery in digital images. In *Proceedings of the Digital Forensic Research Workshop*, 2003.

[GKWB07] T. Gloe, M. Kirchner, A. Winkler, and R. Bohme. Can We Trust Digital Image Forensics? In *Proceedings of the 15th International Conference on Multimedia*, Augsburg, Germany, September 2007.

[Hay01] C. Haythornthwaite. Introduction: The Internet in Everyday Life. In *Proceedings of the American Behavioral Scientist*, 2001.

[HMS+12] M. Hussain, G. Muhammad, S. Q. Saleh, A. M. Mirza, and G. Bebig. Copy-move image forgery detection using multi-resolution weber descriptors. In *Proceedings of the Eighth International Conference on Signal Image Technology and Internet Based Systems*, pages 395–401, 2012.

[Kem20] S. Kemp. Digital Global Overview. https://datareportal.com/reports/digital-2020-global-digital-overview, January 2020.

[LH13] F. Li and T. Huang. Video Copy-Move Forgery Detection and Localization Based on Structural Similarity. https://link.springer.com/chapter/10.1007/978-3-642-41407-7_7, November 2013.

[LWTS07] G. Li, Q. Wu, D. Tu, and S. Sun. A sorted neighborhood approach for detecting duplicated regions in image forgeries based on dwt and svd. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1750–1753, 2007.

[MVP08] A.N. Myrna, M.G. Venkateshmurthy, and C.G. Patil. Detection of region duplication forgery in digital images using wavelets and log-polar mapping. In *Proceedings of*

the Detection of Region Duplication Forgery in Digital Images Using Wavelets and Log-Polar Mapping, pages 371–377, 01 2008.

[RJ16] Y. Rao and Ni J. A Deep Learning Approach to Detection of Splicing and Copy-Move Forgeries in Images. In *Proceedings of the IEEE International Workshop on Information Forensics and Security*, Abu Dhabi, United Arab Emirates, December 2016.

[SM16] K. Sitara and B.M. Mehtre. Digital Video Tampering Detection: An Overview of Passive Techniques. In *Digital Investigation*, September 2016.

[SSB10] B.L. Shivakumar and Lt. Dr. S. Santhosh Baboo. Detecting copy-move forgery in digital images: A survey and analysis of current methods. In *Proceedings of the Global Journal of Computer Science and Technology*, 2010.

[WBSS04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. In *Proceedings of the IEEE Transactions on Image Processing*, 2004.

[WDT09] W. Wang, J. Dong, and Tan T. Effective Image Splicing Detection based on Image Chroma. In *Proceedings of the Effective image splicing detection based on image chroma*, 2009.

[ZGLWT16] Y. Zhanga, J. Goha, L. Lei Wina, and V. Thinga. Image Region Forgery Detection: A Deep Learning Approach. In *Proceedings of the Singapore Cyber-Security Conference*, 2016.