DESARROLLO DE UN ALGORITMO PARA DETECCIÓN DE ESTEGANOGRAFÍA LSB EN IMÁGENES DIGITALES A COLOR USANDO REDES NEURONALES CONVOLUCIONALES

Alberto Manuel García Grimaldos

Universidad Pontificia Bolivariana
Facultad de Ingeniería de Sistemas e Informática
Bucaramanga
2021

DESARROLLO DE UN ALGORITMO PARA DETECCIÓN DE ESTEGANOGRAFÍA LSB EN IMÁGENES DIGITALES A COLOR USANDO REDES NEURONALES CONVOLUCIONALES

Alberto Manuel García Grimaldos

Proyecto de grado como requisito
Para optar el título de
Ingeniero de Sistemas e Informática

Director del Proyecto de Grado
Julián Darío Miranda Calle
Especialista en Seguridad Informática, Ingeniero Electrónico, Ingeniero de Sistemas e Informática

Universidad Pontificia Bolivariana
Facultad de Ingeniería de Sistemas e Informática
Bucaramanga
2021

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

Firma del jurado 1

_____

Firma del jurado 2

Dedicatoria

A todas las personas que están pasando por situaciones difíciles en este momento coyuntural.
A mis hermanos, para recordarles que el esfuerzo siempre llega con una recompensa mayor.
A mi director de proyecto, como recuerdo de su primer trabajo de grado estando del otro lado.
A la Ingeniera Angélica Flórez Abril (QEPD), quién en vida tocó los corazones de cada persona que con ella se relacionó.
A mi suegro y amigo Ismael Enrique Arciniegas Ocampo (QEPD), quién me enseñó sobre la vida, los sacrificios del camino y cómo superar los obstáculos.

Agradecimientos

A la facultad de Ingeniería de Sistemas e Informática y todo su equipo, pues han estado comprometidos con este proyecto desde el primer día; el apoyo incondicional que me han extendido ha sido de gran ayuda para mantenerme en pie a pesar de las diversas situaciones adversas por las que he pasado en este último año.
A Jordan Rodríguez, administrador de los laboratorios, por estar siempre sonriente y disponible para cualquier inquietud técnica en el uso de los recursos del CCA.
A la Ingeniera Johanna Marcela Suárez Pedraza, directora de la facultad, por permitirme ser parte de la familia IngSistemas y por su apoyo en temas personales y académicos.
A mi director de proyecto, Julián Miranda, por ayudarme a encontrarle salidas a cada inconveniente personal y del presente trabajo de grado.
A la Ingeniera Ambiental Lizeth Fernanda Arciniegas Reyes, quien fue un apoyo técnico y moral para el desarrollo de este proyecto.
A mis amigos: los Ingenieros Fabián Barreto y Ciro Gamboa, el candidato a Psicólogo Bayron Suárez y los candidatos a Ingenieros Juan Pablo Plata y Jhodman Contreras; por todos los consejos brindados a lo largo de este proyecto.
A mis maestros José Leonardo Osorio Bravo y Hernán Mejía Borja, por brindarme su apoyo y enseñarme sobre la vida más allá del Tatami y el escenario.
A la Ingeniera Diana Teresa Gómez Forero, por todas las enseñanzas dentro y fuera del aula que ahora llevaré conmigo toda la vida.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ECUACIONES

LISTA DE ANEXOS

LISTA DE ABREVIATURAS

Artificial Intelligence (AI)
Artificial Neural Network (ANN)
Bitmap (BMP)
Break Our Steganographic System! (BOSSbase)
Centro de Computación Avanzada (CCA)
Cover Image Suppression Network-(CIS-Net)
Convolutional Neural Networks (CNN)
Center for Research on Intelligent Perception and Computing (CRIPAC)
Deep Residual learning based Network (DRN)
Exploratory Data Analysis (EDA)
Highpass Low-pass Steganography (HILL)
Highly Undetectable steGO (HUGO)
Integrated Development Environment (IDE)
Joint Photographic Experts Group (JPEG)
Least Significant Bit (LSB)
Peak signal-to-noise ratio (PSNR)
Pixel Value Differencing (PVD)
Recursive Feature Elimination (RFE)
Red, Green, Blue (RGB)
Residual Neural Networks (ResNet)
Secure Shell (SSH)
Single-value Truncation Layer (STL)
Stochastic Gradient Descent (SGD)
Sub-linear Pooling Layer (SPL)
Virtual Private Server (VPS)
Wavelet Obtained Weights steganography (WOW)

Universidad Pontificia Bolivariana
SECCIONAL BUCARAMANGA

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: Desarrollo de un algoritmo para detección de esteganografía LSB en imágenes digitales a color usando redes neuronales convolucionales.

AUTOR(ES): Alberto Manuel García Grimaldos

PROGRAMA: Facultad de Ingeniería de Sistemas e Informática

DIRECTOR(A): Julián Darío Miranda Calle

RESUMEN

En el presente trabajo de investigación, se plantea el desarrollo de un algoritmo mediante redes neuronales convolucionales que permita la detección de esteganografía LSB en imágenes digitales a color, con el fin de coadyuvar a la detección eficiente, precisa y automatizada de esteganografía, mitigar posibles amenazas o ataques de terceros y proteger los sistemas de información y la infraestructura tecnológica de las organizaciones; teniendo en cuenta las limitaciones que se presentan para efectuar estegoanálisis y para detectar visualmente el ofuscamiento de datos o mensajes en las imágenes. El proyecto se lleva a cabo mediante la revisión del estado del arte de investigaciones previas en la detección de esteganografía LSB, seguido por la generación de un conjunto de datos para el entrenamiento y prueba del algoritmo, el diseño del algoritmo de Deep Learning mediante redes neuronales convolucionales y la validación del algoritmo mediante métricas de desempeño. Entre los modelos desarrollados para la detección de esteganografía en imágenes a color, las métricas más elevadas obtenidas fueron: F1 score de 66.00%, tiempo medio de estimación de 55.95 milisegundos y tiempo de entrenamiento de aproximadamente una hora con 15 minutos por epoch. Experimentos adicionales fueron propuestos, de los cuales uno de ellos alcanzó un F1 score de alrededor de 93.00% a partir de un modelo más interpretable y menos complejo en su arquitectura.

PALABRAS CLAVE:

Esteganografía LSB, CNN, Aprendizaje Profundo, Aprendizaje Automatizado, Imágenes RGB.

V° B° DIRECTOR DE TRABAJO DE GRADO

Universidad Pontificia Bolivariana
SECCIONAL BUCARAMANGA

GENERAL SUMMARY OF WORK OF GRADE

TITLE: Development of an algorithm for detection of LSB steganography in digital color images using convolutional neural networks.

AUTHOR(S): Alberto Manuel García Grimaldos

FACULTY: Facultad de Ingeniería de Sistemas e Informática

DIRECTOR: Julián Darío Miranda Calle

ABSTRACT

In the present research work, the development of an algorithm using convolutional neural networks is proposed, which allows the detection of LSB steganography in color digital images, to contribute to the efficient, precise and automated detection of steganography, mitigate possible threats from third parties and protect the information systems and technological infrastructure of organizations; considering the limitations that are presented to make steganalysis, and to visually detect the obfuscation of data or messages in images. The project developed by reviewing the state of the art of previous investigations in the detection of LSB steganography, followed by the generation of a data set for the training and testing of the algorithm, the design of the Deep Learning algorithm through convolutional neural networks and validation using performance metrics. Between the developed algorithms for the detection of steganography in color images, the highest metrics were: F1 score of 66.00%, an average prediction time of 55.95 milliseconds and a training time of around one hour and 15 minutes per epoch. Additional experiments were proposed, of which one of them reached an F1 score of around 93.00% from an interpretable algorithm and less complex in its architecture.

KEYWORDS:

LSB Steganography, CNN, Deep Learning, Machine Learning, RGB images.

V° B° DIRECTOR OF GRADUATE WORK

INTRODUCCIÓN

A medida que ha avanzado la tecnología, la humanidad se ha beneficiado de sus numerosos avances en ámbitos como la medicina, la ciencia y la comunicación. Sin embargo, así como la tecnología puede usarse para mejorar procesos y automatizar procedimientos repetitivos, también puede ser utilizada como medio de perjuicio [1]–[3]. Con el avance de la tecnología, las personas y organizaciones con malas intenciones han encontrado medios para realizar sus actividades y perjudicar a otros de forma casi imperceptible. Uno de estos medios ha sido la esteganografía.

La esteganografía es un método que permite el ofuscamiento de datos o mensajes en archivos multimedia, como imágenes, audios, y documentos de ofimática, entre otros. Uno de los medios más usados para embeber esteganografía son las imágenes digitales, ya que es más fácilmente transmisible, se hace imperceptible la detección por medio de la visión humana y se requieren diferentes técnicas que implican análisis profundos para su identificación, como el estegoanálisis, que en ocasiones resulta ser complejo de efectuar por analistas de seguridad (estego-analistas).

Las redes neuronales convolucionales (CNN) son arquitecturas de Deep Learning que imitan las neuronas de la corteza visual primaria del ojo humano con el fin de identificar las características de las imágenes, por lo cual son ideales para desarrollar algoritmos para el estegoanálisis.

Este documento consta de una revisión del estado del arte, donde se abordan las diferentes técnicas empleadas para la creación de redes neuronales convolucionales aplicadas al estegoanálisis, seguida por la generación del conjunto de datos para el entrenamiento y prueba del algoritmo de Deep Learning, el diseño del algoritmo mediante redes neuronales convolucionales y la validación de los algoritmos mediante la proposición de métricas de desempeño especializadas.

1 PLANTEAMIENTO DEL PROBLEMA

La esteganografía es un método para el ofuscamiento de datos (o archivos digitales) ocasionalmente maliciosos en objetos portadores, que busca aprovechar las limitaciones biológicas del sistema de visión humano, entre otros, para ser difícilmente identificable [4]. Según McAfee, “el uso de esteganografía en ciberataques es fácil de implementar y difícil de detectar”; siendo que casi cualquier tipo de archivo se puede embeber en otro, hay muchos métodos para atacar un sistema informático [5]. Adicionalmente, la manera habitual [6] de detectar esteganografía en imágenes es mediante estegoanálisis, procedimiento mediante el cual se detectan y revelan mensajes que han sido ocultados por medio de esteganografía [7]. Sin embargo, tal como lo establece K. Karampidis, et. al. [4], este método tiene sus limitaciones, ya que un analista de seguridad podrá ser capaz de analizar un reducido número de imágenes al día [8].

La esteganografía LSB (Least Significant Bit), es una técnica de esteganografía ampliamente utilizada, que consiste en reemplazar los bits menos significativos de un pixel de una imagen con bits de una imagen oculta o un mensaje secreto, permitiendo transmitir grandes cantidades información de manera imperceptible al ojo humano, que es poco sensible a variaciones de un bit en la intensidad de color; [9] a más bits reemplazados habrá menor imperceptibilidad, pero mayor capacidad de ocultamiento.

El inconveniente de no protegerse contra la esteganografía recae en que esta puede ser perjudicial para un sistema de información o infraestructura tecnológica, ya que se podría ejecutar maliciosamente contenido embebido sin ser percibido como amenaza por las medidas de protección [9]–[12]. Por lo tanto, se hace necesario el desarrollo de estrategias que faciliten la detección automatizada de la esteganografía en imágenes monocromáticas y a color, ya que en ocasiones el estegoanálisis puede resultar dispendioso y complejo de efectuar por los analistas de seguridad.

2 OBJETIVOS

2.1 Objetivo general

Desarrollar un algoritmo para la detección de esteganografía LSB en imágenes digitales a color utilizando redes neuronales convolucionales, con el fin de coadyuvar en el proceso de detección automatizada de esteganografía LSB.

2.1.1 Objetivos Específicos

- Analizar las características de las técnicas de redes neuronales convolucionales aplicadas a la detección de esteganografía LSB mediante la revisión del estado del arte, tal que permita la caracterización de los datos de entrada y salida requeridos para el proceso de detección.

- Generar el conjunto de datos para el entrenamiento y prueba del algoritmo mediante técnicas de preprocesamiento de datos a partir de diversas fuentes de imágenes digitales a color.

- Diseñar un algoritmo de Deep Learning para detección de esteganografía LSB en imágenes digitales a color que implemente redes neuronales convolucionales.

- Validar el algoritmo para obtener métricas de desempeño en la detección de esteganografía LSB, mediante métodos de validación de desempeño y eficiencia algorítmica propios del Deep Learning.

3 JUSTIFICACIÓN

Teniendo en cuenta la creciente necesidad de proteger sistemas de información contra todo tipo de amenazas, se hace evidente la necesidad de reforzar los sistemas frente a vectores de ataque. Si bien los sistemas de información se protegen contra esteganografía en imágenes (por medio del uso servicios de terceros especializados en seguridad informática, control de acceso a los servicios tecnológicos propios y análisis comportamental dinámico en un entorno controlado de los archivos al ser ejecutados) [9] [13], hay diferentes tipos que, de ser eficaces en su intrusión, podrían llevar incluso a la ejecución remota de código en un servidor que se cree protegido [14]; sin embargo, la limitante no es solo tecnológica, ya que las condiciones biológicas del ser humano y la naturaleza de la visión no dan cabida a la detección visual de la esteganografía en imágenes.

Pasar por alto la existencia de esteganografía en un archivo es común, ya que percatarse de esto conlleva una serie de actividades de estegoanálisis que, por lo general, requerirán disponer de un archivo portador [15], [16]. Dependiendo del tipo de esteganografía, se pueden almacenar instrucciones para ejecución de scripts o comandos nativos en el sistema operativo gracias al almacenamiento del archivo malicioso, que podría incluso llegar a comprometer seriamente la seguridad de una organización [17], [18].

Dado que la detección manual de esteganografía LSB en imágenes digitales a color requiere de técnicas que podrían llegar a ser complejas de escalar si se obtuviera un volumen mayor de imágenes a analizar, se hace necesario encontrar una forma de automatizar este procedimiento sin comprometer recursos espaciales y temporales, los cuales ya son escasos para las técnicas actuales pues estas son costosas computacionalmente comparadas con la baja precisión que otorgan [19], [20]. Por esta razón, en el presente trabajo de grado se propone el desarrollo de un algoritmo mediante redes neuronales convolucionales que permita la detección automatizada de esteganografía LSB en imágenes digitales a color.

4 ESTADO DEL ARTE

Un estudio de antecedentes es realizado para establecer el estado actual de las investigaciones relacionadas con detección de esteganografía LSB en imágenes digitales a color utilizando redes neuronales convolucionales, lo que permite identificar fechas recientes para el uso de la técnica objeto del estudio y así enmarcar la investigación como una temática de novedad e interés.

En 2016, los investigadores de CRIPAC (Center for Research on Intelligent Perception and Computing) [21] desarrollaron una CNN que para aprender los atributos de imágenes cargadas con los algoritmos esteganográficos WOW y S-UNIWARD utilizando payloads altos, que posteriormente serviría para detectar imágenes cargadas con payloads más bajos. Se utilizó la base de datos de la competencia BOSS con 10,000 imágenes monocromáticas de dimensión 512x512, con una distribución de 50/50 entre imágenes portadoras y cargadas con esteganografía. La asignación del conjunto de datos fue de 70/10/20 (entrenamiento, validación y prueba) para el conjunto de datos, para el cual se tomaron payloads de 0.1, 0.2, 0.3, 0.4 y 0.5 bits por píxel (bpp). El desempeño del estudio arroja una precisión promedio de entre 84.00% y 86.00%.

También en 2016, Couchot, et. al.[22], utilizaron una red neuronal convolucional (CNN) para detectar esteganografía embebida con los siguientes algoritmos: HUGO (Highly Undetectable steGO), WOW (Wavelet Obtained Weights steganography), S-UNIWARD, STABYLO, EAI-LSBM, y MVG; se buscaba optimizar el proceso utilizando filtros más grandes y menos convoluciones, para de esta forma poder sortear las dificultades computacionales de trabajar con imágenes de mayor resolución y menor payload. Se usó la ya mencionada base de datos BOSSbase y se añadió la carga esteganográfica con los algoritmos WOW, HUGO, y J-UNIWARD utilizando cargas de 0.4 y 0.1 bpp, alcanzando así una precisión de entre 70.89% y 97.09%.

Wu, et. al. [15], en el 2016, llevaron a cabo una investigación de una red basada en aprendizaje profundo residual (en inglés Deep Residual learning based Network) o DRN como modelo de CNN para estegoanálisis. Se hizo uso de la versión 1.01 del dataset BOSSbase, que contiene 10.000 imágenes naturales en escala de grises de 512x512, de las cuales, mediante data augmentation (aumento de datos), se obtuvieron 40,000 imágenes de 256x256. Se comparó el rendimiento del modelo con cinco algoritmos de última generación: HUGO, WOW, S-UNIWARD, HILL (Highpass Low-pass steganography) y MiPOD. Se obtuvo como resultado que este modelo tiene tasas de error entre 4.10% y 10.40%.

Siguiendo con la investigación anterior, Wu, et. al. [16], propusieron en 2018 un modelo de CNN para estegoanálisis basado en aprendizaje residual. La DRN propuesta por los autores cuenta con dos propiedades importantes: tiene una gran cantidad de capas de red y conserva el esteganograma (la información secreta oculta dentro de la imagen). Como conjunto de datos se usó BOSSbase V1.01 con la carga esteganográfica del algoritmo S-UNIWARD con payloads de 0.4 bpp. El modelo DRN permitió detectar algoritmos esteganográficos con un error de entre 4.30% y 10.40%.

Wu, et. al., en 2019 [19], plantearon un modelo llamado Cover Image Suppression Network (CIS-Net), que mejoró el rendimiento del estegoanálisis en imágenes espaciales, mediante la supresión del contenido de la imagen portadora, usando dos capas: STL o Single-value Truncation Layer y SPL o Sub-linear Pooling Layer. Fueron usadas imágenes del conjunto de datos empleado en su investigación anterior y Matlab como entorno de desarrollo. Para evaluar el desempeño, se escogieron tres algoritmos representativos: WOW, S-UNIWARD y HILL. Como resultados, obtuvo que STL permite reducir la variación de las características de entrada sin deteriorar información útil y SPL, utilizando una función de potencia sub-lineal, permitió suprimir elementos de gran valor en el contenido de la imagen portada. Se usaron payloads de 0.1 a 0.5 bpp y se obtuvo un mejor rendimiento en general (porcentaje de error desde 36.82% a 10.73%) sobre el modelo ReST-Net (modelo para transferencia de aprendizaje).

En el mismo año 2019, Butora, et. al., [20] construyeron herramientas de estegoanálisis para detectar algoritmos de incrustación y métodos esteganográficos mediante redes neuronales convolucionales residuales profundas, haciendo uso de la arquitectura SRNet con imágenes monocromáticas de 256x256 de las databases BOSSbase 1.01 y BOWS2, cada una con 10,000 imágenes. Las 10,000 imágenes de BOSSbase fueron divididas en tres conjuntos: el primero de 4,000, el segundo de 1,000 y el tercero de 5,000. Para las imágenes de entrenamiento se unieron BOWS2 y el primer conjunto, obteniendo un total de 14,000 imágenes; los otros conjuntos fueron usados para validación y pruebas. Fueron estudiados tres métodos: clasificador binario (binary classifier), detector multiclase (multi-class detector) y detector de cubos (bucket detector). Los porcentajes de error obtenidos van entre 3.30% y 33.30%.

También en el año 2019, Kang, et. al.[23], estudiaron un método de estegoanálisis basado en CNN que permite una clasificación ternaria (de imágenes de portada, imágenes de WOW con estego e imágenes de UNIWARD con estego) para identificar simultáneamente los algoritmos de esteganografía de imágenes adaptativas representativas WOW y UNIWARD, teniendo en cuenta que estos tienen similares métodos de incrustación de mensajes, que permiten medir y minimizar el grado de distorsión de las imágenes, lo cual hace que muchas veces sea difícil distinguir entre ambos. Los experimentos fueron realizados usando imágenes en escala de grises de 512x512 del database BOSSBase 1.01, con un payload aleatorio de 0.4. Las CNN se implementaron usando la biblioteca TensorFlow y se usaron 16 filtros Gabor (que permiten extraer características relevantes en distintas escalas y direcciones, usados frecuentemente para identificar texturas) y 10 filtros SRM (modelo espacialmente rico). Este método puede efectuar la clasificación ternaria con una precisión de aproximadamente 72.00%.

En complemento, Mamada [24] desarrolló una propuesta para realizar estegoanálisis mediante redes neuronales convolucionales con imágenes digitales a color obtenidas de la base de datos ImageNet, y modelos pre-entrenados, como lo son SE-ResNeXt-50 y SE-ResNeXt-101. Las 962 imágenes de entrada fueron giradas y recortadas aleatoriamente como proceso de *data augmentation* (proceso por el cual se realiza generación de datos artificiales para hacer invariante el resultado arrojado por el algoritmo ante transformaciones espaciales) [25]. Se usó la biblioteca de Deep Learning Chainer para experimentos y la herramienta StegDetect para identificar el contenido esteganográfico. Se obtuvo valores de 66% en F1 score, 50.00% de precisión y 94.00% en recall score.

Kim et. al. [26], propusieron un método para estegoanálisis basado en CNN usando dos imágenes de entrada: la imagen original y su imagen con esteganografía y datos adicionales incrustados, con el objetivo de mejorar el rendimiento; además, dos variantes de CNN convencionales para estegoanálisis: CNN de doble canal y CNN de doble red. Fueron usadas 10,000 imágenes del database BOSSBase 1.01 divididas en cuartos, para obtener finalmente 40,000 imágenes. Los esteganogramas fueron generadas usando el método S-UNIWARD con diferentes payloads. Se hizo uso de la biblioteca TensorFlow y los hiperparámetros se ajustaron teniendo en cuenta métodos convencionales. La precisión reportada fue de 82.00%.

Wang, et. al. [27] propusieron en su investigación del 2020 un algoritmo de estegoanálisis con CNN que utiliza un mecanismo de detección de dominio conjunto, aplicando filtros de paso alto del SRM y patrones residuales de transformada del coseno discreta (DCTR) y un mecanismo de detección no lineal basado en SRM con un método de adquisición de características residuales no lineales. Además, aplicaron un nuevo método de aprendizaje por transferencia para mejorar el rendimiento del modelo. Para el entrenamiento y validación utilizaron las imágenes de los datasets BOSSBase 1.01, escaladas a 256x256 mediante Matlab. Para la prueba del modelo utilizaron los algoritmos WOW y S-UNIWARD, cada uno con payloads 0.2 y 0.4 bpp, obteniendo en el mecanismo de detección no lineal una mayor precisión, de entre 0.30% y 6.00%, en comparación con modelos básicos y en el mecanismo de detección de dominio conjunto, una precisión entre 2.00% y 3.00% mayor. El método de aprendizaje mostró mayores niveles de precisión que otros modelos, como Zhu-Net, obteniendo una precisión de 77.70% a 92.00%.

Chen, et. al.[28], en el 2020, propusieron una red neuronal convolucional adaptativa ligera llamada IAS-CNN, mediante el método de filtro de autoaprendizaje en la red, que permite reducir el consumo de recursos y mejorar la velocidad de procesamiento. La red propuesta cuenta con tres capas: una de preprocesamiento, una de extracción de características y una de clasificación. El dataset usado fue BOSSBase v1.0119, que contenía 10,000 imágenes monocromáticas de 512x512, que posteriormente fueron escaladas a un tamaño de 256x256. Generaron imágenes esteganográficas de diferentes algoritmos y mapas de probabilidad de incrustación de imágenes de portada. Para evaluar el rendimiento de la red, fueron usados los algoritmos HUGO, WOW y S-UNIWARD, cada uno con tres payloads de 0.2, 0.4 y 1.0 bpp, encontrando que el rendimiento de IAS-CNN aumenta a medida que aumenta el payload. Se concluyó que funciona bien para el estegoanálisis, permitiendo obtener un alto rendimiento con menos parámetros y cálculos, con un desempeño final de entre 62.40% y 93.20%.

Miranda, et. al. [29], en el 2020, propusieron una red neuronal artificial con enfoque de atributos, basándose en los resultados obtenidos por Miranda [30] en el año anterior, donde se estructuró un dataset de atributos para imágenes monocromáticas de 256x256. Organizadas por tuplas (portadora e imagen con carga esteganográfica LSB) con variaciones entre 0.1 y 0.5 de payload, se obtuvo un total de 70,000 observaciones. Los resultados de del modelo propuesto fueron precisiones entre 91.45% para detección en general y 96.78% para detección individual por payload.

En el presente año, Kang, et. al. [31], propusieron un modelo ResNet para detectar e identificar diferentes tipos de esteganografía: LSB, PVD (Pixel Value Differencing), WOW y S-UNIWARD, obteniendo resultados de 79.71% en precisión, la cual encontraron un 23.00% superior a la

obtenida al implementer un modelo estándar de CNN. Utilizaron una variación del dataset BossBase 1.01, en la cual extrajeron cuatro ventanas por imagen, para un total de 40,000 observaciones, aplicando payloads de 0.4. El modelo inicial propuesto fue basado en XuNet [32], con el cual comprobaron que no se pueden identificar los diferentes algoritmos esteganográficos con una CNN, llegando al resultado de una red que realiza los siguientes pasos: ResNet para clasificación multiclase entre imágenes portadoras y embebidas para LSB, PVD y tupla de WOW y S-UNIWARD. En el siguiente paso, una ResNet clasifica de forma binaria entre WOW y S-UNIWARD, debido a que encontraron dichos algoritmos mucho más difíciles de detectar.

Como condensado del Estado del Arte se presenta la *Tabla 1*, donde se recogen los resultados obtenidos por anteriores investigadores. Cabe resaltar la poca documentación existente de hiperparámetros y datasets. Algunos autores mencionan el dataset usado, pero no la distribución natural de sus imágenes, lo que podría ocasionar sesgo y falta de generalización de los modelos planteados, pues, al no tener en cuenta el balance natural de clases, es probable que caiga el rendimiento del modelo en una clase desbalanceada.

Tabla 1. Condensado del estado del arte.

Referente	**Métrica**				
	Tasa de error	**Payload (bpp)**	**Precisión**	**F1 score**	**Recall**
[15]	4.10% - 10.40%	--	--	--	--
[16]	4.30% - 10.40%	0.4	--	--	--
[19]	10.73% -36.82%	0.1 – 0.5	--	--	--
[21]	--	0.1 – 0.5	84.00% - 86.00%	--	--
[22]	--	0.1 y 0.4	70.89% y 97.09%	--	--
[20]	3.30% - 33.30%	--	--	--	--
[23]	--	0.4	72.00%	--	--
[26]	--	--	82.00%	--	--
[24]	--	--	50.00%	66.00%	94.00%
[27]	--	0.2 y 0.4	77.70% - 92.00%	--	--
[28]	--	0.2, 0.4 y 1.0	62.40% - 93.20%	--	--
[29]	--	0.1 – 0.5	93.77%	91.22%	88.97%
[31]	--	0.4	79.71%	--	--

Fuente: Autor.

5 MARCO TEÓRICO

El marco teórico, se divide en cuatro temáticas de estudio, consideradas relevantes para el planteamiento de la propuesta: esteganografía, Aprendizaje de Máquina (Machine Learning), Aprendizaje Profundo (Deep Learning) y metodología ágil de desarrollo de software Scrum.

5.1 Esteganografía

La palabra esteganografía proviene del griego steganos (oculto) y graphos (escritura) y significa escritura encubierta [33]. Es una disciplina que estudia las técnicas para el ofuscamiento de información, que puede encontrarse en dispositivos o mensajes, en ficheros contenedores o en contenidos multimedia, como imágenes digitales, audio y videos, de tal forma que pase inadvertida para terceros y solo pueda ser detectada por una persona legítima [34].

Aunque la esteganografía en ocasiones se use con fines didácticos o para proteger información, en muchos casos es usada por cibercriminales para fines de espionaje, robo de datos o infectar dispositivos con malware, e incluso terroristas, para comunicarse entre ellos sin ser descubiertos [35] [17] [36].

La esteganografía suele usarse, sobre todo, en imágenes digitales, ya que tienen un tamaño superior al de otros formatos, como archivos de texto, lo cual permite guardar una gran cantidad de datos. En el modelo de color RGB (rojo, verde y azul), las imágenes ocupan 24 bits de memoria por píxel; si 1 o 3 bits tienen información oculta, los cambios en la imagen son imperceptibles para el ojo humano [37] [38] [39].

Entre las técnicas más usadas de esteganografía está la inserción del bit menos significativo o LSB, que consiste en sustituir el bit menos significativo de cada uno de los bits de un mensaje original por cada uno de los bits del mensaje oculto [14].

El método de esteganografía LSB (Least Significant Bit), consiste en tomar los bits menos significativos del valor binario propio de un píxel perteneciente a una imagen y reemplazarlos por bits de una imagen secreta o mensaje para ocultar información de forma que sea imperceptible para el ojo humano [40]. Es posible conocer el mensaje oculto comparando la imagen inicial con la imagen con contenido embebido. Para determinar la cantidad de bits a reemplazar, se debe tener en cuenta que, a mayor cantidad de bits reemplazados, habrá menor imperceptibilidad, pero mayor capacidad de ocultamiento [41].

Teniendo una profundidad de color de 2 bits, por ejemplo, el bit menos significativo puede ser alterado, pero esto daría como resultado una corta paleta de colores disponibles, ya que al ser menor la cantidad de bits utilizada para la resolución de colores, menor cantidad de estos se podrán representar. Por el lado contrario, al comparar dos colores con el último bit alterado en una representación de 8 bits, se podrán notar sutiles diferencias si observa con detenimiento; a mayor tamaño de imagen y mayor profundidad de color, mayor dificultad en la detección visual

de esta alteración intencionada. Debido a lo anterior, se dice que la esteganografía LSB es imperceptible al ojo humano [42].

En la *Figura 1*, se puede evidenciar el cambio de color progresivo del recuadro a la izquierda a medida que se altera desde el bit menos significativo al más significativo del segmento azul en la paleta RGB.

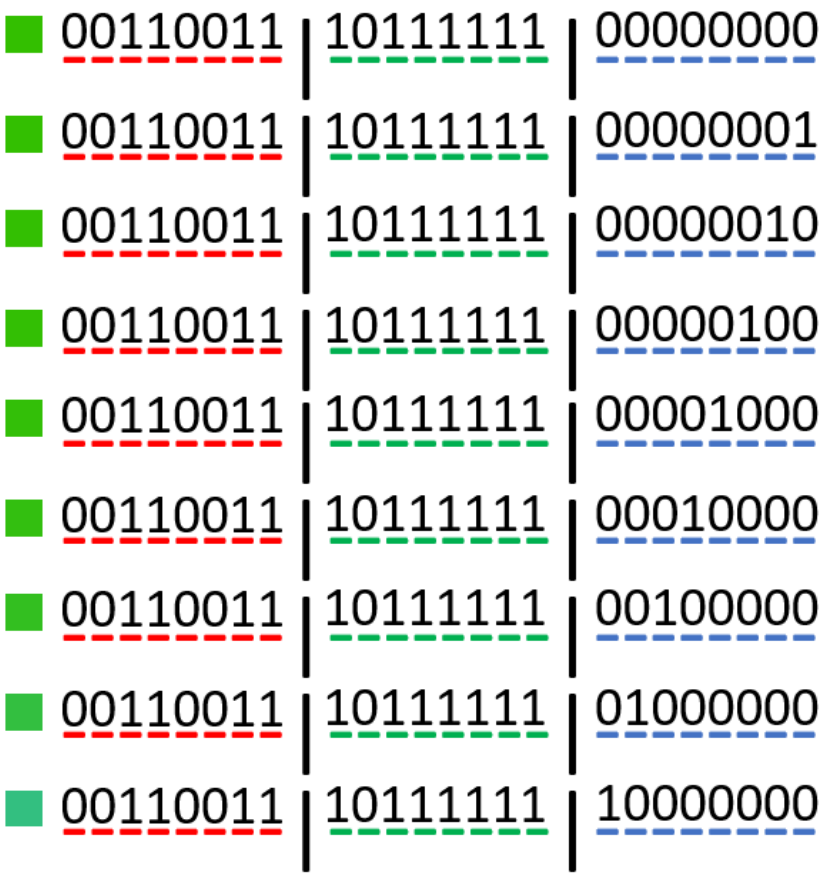

Figura 1. Cambio de color progresivo. Fuente: Autor.

En las *Figuras 2 y 3*, se muestra cómo el ojo humano es incapaz de percibir cambios sutiles en la escala de colores de un píxel a otro, utilizando un ejemplo práctico. La *Figura 2*, muestra un cambio de color de 31 intensidades del color azul y la *Figura 3*, muestra que las dos primeras filas tienen colores aleatorios con alteraciones de hasta el bit más significativo, las siguientes 3 tienen el color inicial aumentado en 4 niveles en la intensidad de color respecto a la fila anterior y la última fila tiene el color original.

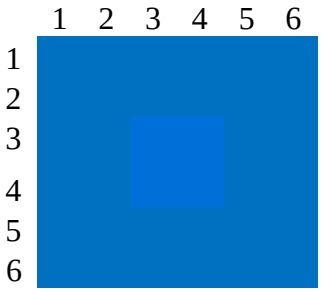

Figura 2. Cambio de color de 31 intensidades de color. Fuente: Autor.

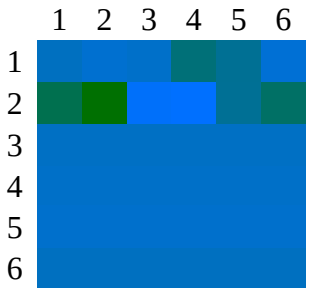

Figura 3. Alteraciones de color de hasta el bit más significativo (filas 1 y 2), 4 niveles de intensidad de color (filas 3, 4 y 5) y color original (fila 6). Fuente: Autor.

Con base en el ejemplo anterior, se puede determinar gráficamente que la variación mínima en el color de un píxel es imperceptible al ojo humano, sentando así las bases para justificar la presente investigación. Según Béla Török [43], los valores mínimos de Delta E (CIE Lab) para que el ojo humano perciba una diferencia en el color es de 0.50 - 1.00 para observadores expertos, 1.00 – 2.00 para obtener una diferencia mínima, 2.0 – 4.0 para obtener una diferencia perceptible, 4.00 – 5.00 para una diferencia significativa y mayor a 5.00 para obtener un color diferente.

5.2 Aprendizaje de Máquinas (Machine Learning)

El Aprendizaje de Máquinas o Machine Learning es una rama de la Inteligencia Artificial que se encarga de imitar la forma en la que los seres humanos y animales aprenden, con el objetivo de realizar análisis de datos complejos con poca o nula intervención humana. Mediante métodos computacionales, los algoritmos aprenden información a partir de los datos suministrados y se van adaptando a medida que los datos aumentan en volumen, para estimar datos futuros [44]. Puede llevarse a cabo por medio diferentes técnicas:

- Aprendizaje supervisado, es aquel en el que se usan conjuntos de datos de entrada y salida conocidos y etiquetados, para entrenar algoritmos y crear modelos que clasifiquen datos o estimen resultados con precisión [45]. Es ampliamente utilizada en organizaciones, por ejemplo, para determinar el correo no deseado [46].

- Aprendizaje no supervisado, este tipo de aprendizaje explora los datos y establece patrones en datos de entrada no etiquetados [47], siendo así útil para segmentación de clientes y detección de anomalías en conjuntos de datos crecientes a lo largo del tiempo [48].

- Aprendizaje semisupervisado, este aprendizaje acepta como entrada, generalmente, pocos datos etiquetados y muchos datos no etiquetados para el entrenamiento, con el objetivo de que los datos etiquetados orienten la clasificación de los no etiquetados [49].

- Aprendizaje por refuerzo, el cual se encarga de que las máquinas determinen, sobre la marcha, comportamientos a partir de la búsqueda de errores y recompensas. La mejor solución o recomendación es presentada por un refuerzo de resultados exitosos [50]. Este tipo de aprendizaje suele ser usado en el desarrollo de ambientes de videojuegos [51].

A continuación, se hará una profundización sobre el aprendizaje supervisado, ya que ha sido el utilizado para el desarrollo del algoritmo propuesto en esta investigación.

Dentro de los modelos de aprendizaje de máquina, se encuentran las técnicas de clasificación, estas se encargan de estimar respuestas discretas, clasificando los datos de entrada a partir de descriptores y su relación con la variable. Se usan si los datos se pueden etiquetar, categorizar o separar en grupos o tipos. Es usualmente usado en reconocimiento de voz y escritura, imágenes médicas y detección de fraudes de identidad [52].

En las técnicas de clasificación binaria, generalmente se usa una función umbral (threshold) para devolver un valor binario de la clasificación o, en su defecto, un 0 para falso y 1 para verdadero, como se puede ver en la ecuación 1.

$$\text{Tag} = \text{y}_{\text{pred}} > 0.50$$ (Ecuación 1. Función de umbral para clasificación binaria.)

donde,

$Tag = categoria\ clasificada$
$y_{pred} = categoria\ predicha\ por\ el\ modelo$

Dentro de los algoritmos de clasificación más usados en la actualidad están: la regresión logística y las redes neuronales artificiales.

Regresión logística: es un modelo de análisis predictivo propuesto usualmente cuando la variable dependiente es binaria. Se utiliza para describir cómo cambia una variable dependiente a medida que cambian las variables independientes o predictoras. Es ampliamente usada en el análisis de datos en medicina, epidemiología y en situaciones en las que se requiera determinar la presencia o ausencia de un evento, debido a que es uno de los modelos más interpretables y explicables. A continuación, se presenta su función [47] [54]. En la ecuación 2 se muestra la forma general de la función de regresión:

$y = w_0 + w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n$ (Ecuación 2. Función de regresión. Fuente:[55])

donde,

y = Variable dependiente
w_n = Pesos
w_0 = Bias
x = Variable predictora

Esta regresión es ingresada a una función de activación sigmoide que permite obtener una salida que puede interpretarse como una probabilidad:

$\sigma(y) = \frac{1}{1+e^{-y}}$ (Ecuación 3. Función de activación sigmoide. Fuente: [56])

donde,

y = Variable dependiente
σ = Función sigmoide

Redes Neuronales Artificiales (ANN): son modelos computacionales que intentan imitar el comportamiento del cerebro humano, específicamente, del sistema nervioso. Están conformadas por neuronas artificiales o nodos que están conectados entre sí y reciben información. Para que la red aprenda, se usa un conjunto de observaciones de las variables como entrenamiento. Su objetivo es estimar un valor de salida a partir de valores de entrada o atributos. Una red neuronal artificial está conformada por un conjunto de entrada, donde los enlaces reciben información de afuera o de otras neuronas, un conjunto de salida, que entrega el resultado de la red neuronal, puede ser numérico o binario, generalmente establecido con un límite para aproximar a 1 o 0, dependiendo de la variable a obtener resultado y funciones, que se presentan a continuación:

- Función de propagación: relaciona los valores de entradas, sus pesos y un sesgo o bias para calcular la salida.

- Función de activación: determina el valor de salida de una neurona, computando los pesos con los datos ingresados y realizando una modificación de acuerdo con la función.

- Función de transferencia: modela la relación entre la entrada y la salida.

El sistema de redes neuronales artificiales está dividido en tres tipos de capa, como se muestra en la *Figura 4:* capa de entrada (datos de entrada), capa oculta (procesamiento de información) y capa de salida (datos de salida) [57].

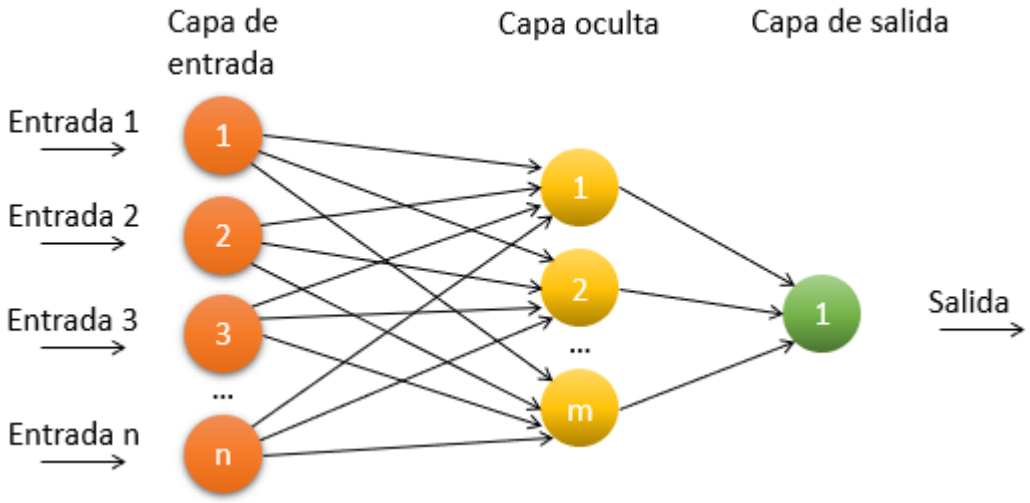

Figura 4. Capas neuronales de un sistema de Aprendizaje de Máquina. Adaptado de: [58].

5.2.1 Validación de modelos de Machine Learning

La validación de modelos es un proceso en el cual se lleva a cabo una serie determinada de análisis para verificar la veracidad de una estimación con respecto a los datos ya existentes.

Las técnicas empleadas para determinar la capacidad predictiva (capacidad de estimación) del modelo creado pueden ser estándar (utiliza el conjunto de entrenamiento para realizar la validación del modelo entre el valor estimado y el valor real) o de validación cruzada (utiliza datos no presentados al modelo durante el entrenamiento, para así poder verificar el funcionamiento bajo una posible situación del mundo real).

Existen numerosas técnicas de validación cruzada de un modelo, algunas de las cuales serán explicadas a continuación:

- Muestreo aleatorio: Consiste en dividir el conjunto de datos al azar manteniendo el balance establecido entre las variables objetivo. Se suele dividir en 70/20/10 (70% entrenamiento, 20% prueba y 10% validación) y 80/20 (80% entrenamiento, 20% prueba) [59].

- Muestreo estratificado: Consiste en dividir el conjunto de datos manteniendo la proporción de las categorías de una variable o las clases de la variable objetivo, de tal forma que se mantenga el desbalance de clases del conjunto de datos original previo a la división [60].

- *k*-Fold Cross-Validation: Esta técnica de validación consiste en dividir el conjunto de datos en *k* número de subconjuntos, usar un subconjunto de prueba y entrenar los datos con *k-1* iteraciones, cambiando en cada iteración el conjunto de pruebas. Este proceso se repite en un número *k* de iteraciones y se promedia el resultado en cada una de estas [61].

5.2.2 Optimización de hiperparámetros

Los hiperparámetros se definen como los valores establecidos manual o automáticamente por los usuarios o al entrenar modelos (conocidos informalmente como los parámetros de los parámetros); estos se usan para estipular los parámetros del modelo que se aprenden o especifican y son indispensables para llevar a cabo las estimaciones [62].

La optimización de los hiperparámetros en el análisis de datos se realiza para obtener la configuración de valores óptima de cada atributo para que el modelo tenga el mejor rendimiento posible con el menor costo computacional, como se observa en la *figura 5*.

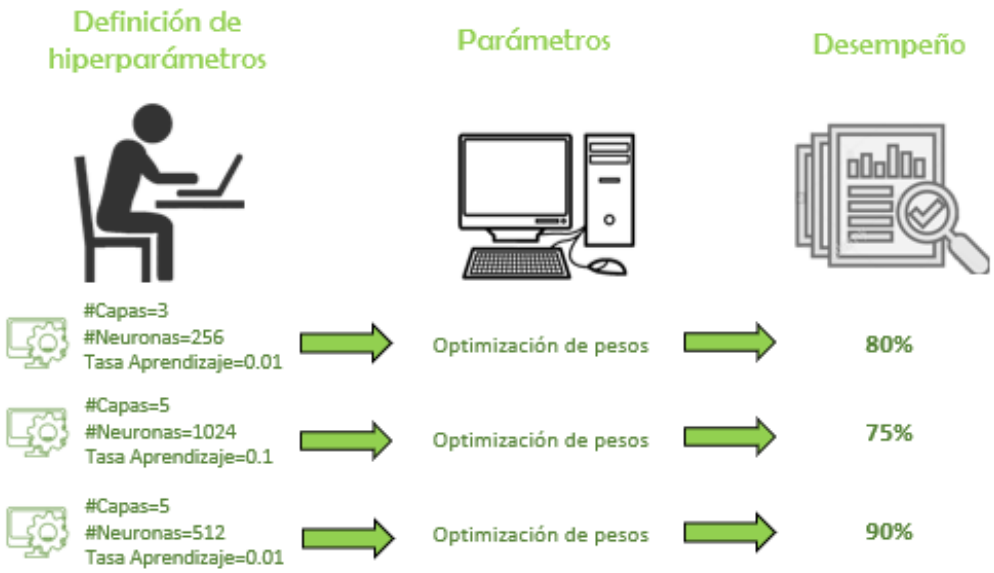

Figura 5. Proceso de optimización de hiperparámetros. Fuente: Autor

Existen diferentes métodos para la determinación de esta configuración:

- Búsqueda manual: como su nombre lo indica, este proceso lo realiza el analista de manera manual, incorporando el conocimiento sobre cómo ciertos cambios de hiperparámetros pueden afectar el desempeño del modelo. Esta técnica es la más común, ya que consume menos tiempo de procesamiento [63].

- Búsqueda en cuadrícula: en este procedimiento, el modelo se encarga de determinar, mediante el recorrido de todos los puntos del espacio establecido de búsqueda, la combinación que otorguen un mejor resultado final en el modelo. Este proceso resulta ser

más intensivo computacionalmente, pues evalúa todos los puntos en un espacio determinado [64].

- Búsqueda aleatoria: a diferencia de la búsqueda en cuadrícula, este método se encarga de optimizar los hiperparámetros recorriendo el espacio de búsqueda aleatoriamente, en lugar de realizarlo para todo el dominio del espacio seleccionado [64].

- Ajuste automático de hiperparámetros: entre las que se encuentran la optimización Bayesiana, el Descenso de Gradiente y los Algoritmos Evolutivos, se encargan de realizar la optimización basándose en la función pérdida del resultado obtenido por anteriores evaluaciones, disminuyendo el tiempo computacional empleado [65] [66].

5.3 Aprendizaje Profundo o Deep Learning

El Aprendizaje Profundo o Deep Learning es una técnica que hace parte de la Inteligencia Artificial (AI), estructurada con el fin de emular el comportamiento de las profundas redes neuronales del cerebro humano, así persiguiendo obtener un tipo de aprendizaje intuitivo propio de los seres humanos. Esto se realiza entrenando al sistema para que revise datos, reconozca patrones y realice estimaciones de comportamientos futuros, clasificaciones o descripciones, detecte objetos y reconozca el habla, entre otras funciones. Al ser un sistema más complejo, se presentan varias ventajas sobre el aprendizaje tradicional de las máquinas, como mayor exactitud, rapidez y que permite omitir el proceso de extracción de atributos; los algoritmos en el Aprendizaje Profundo muestran como salida un modelo estadístico, aplicando una transformación lineal como entrada [67]. Entre las desventajas del Aprendizaje Profundo están la gran cantidad de datos que requiere, costos computacionales más elevados debido al procesamiento de esos datos y el difícil entendimiento de cómo funciona un modelo tomando decisiones (extremadamente baja interpretabilidad y explicabilidad) [68].

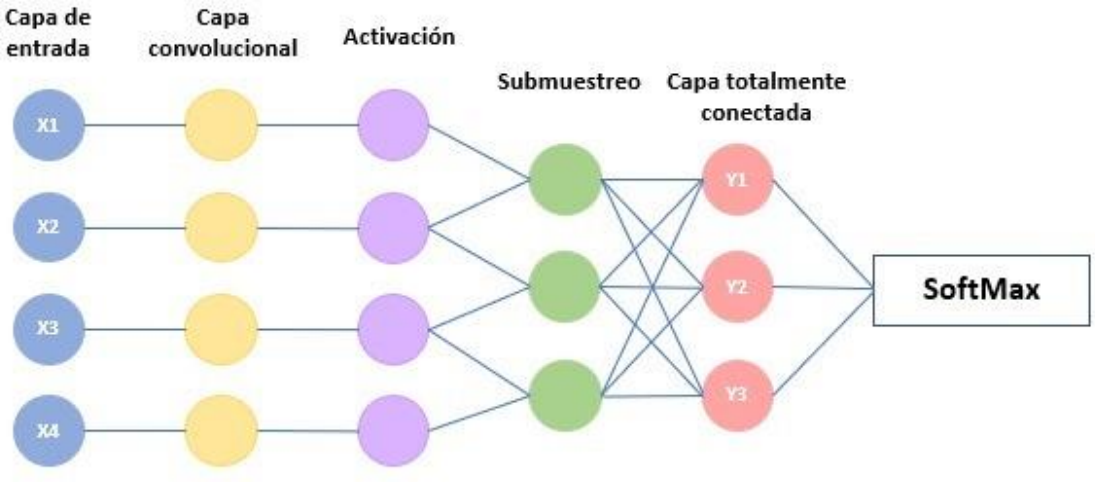

Figura 6. Capas neuronales de un sistema de Aprendizaje Profundo. Fuente: Autor.

Algunas de las aplicaciones del Deep Learning en la actualidad se presentan a continuación:

- Deep Learning en la agricultura: en la agricultura inteligente, el Deep Learning ha permitido el diseño de algoritmos que permitan contrarrestar efectos producidos por el cambio climático, como lo son los problemas con el suministro de agua. El cambio climático ha ocasionado que las estaciones secas y húmedas no sean igual de predecibles que antes, por lo cual es indispensable el seguimiento y evaluación en tiempo real del estado del clima; esto termina generando grandes cantidades de datos que es posible analizar mediante el desarrollo de redes neuronales convolucionales [69].

- Deep Learning en la ecología: en la ecología, unos de los mayores desafíos son la gestión y conservación de la diversidad biológica. El Deep Learning ha permitido abordar una gran cantidad de elementos en este ámbito, como lo son la identificación de plantas y animales salvajes en imágenes, el fenotipado de plantas, la detección de afecciones en los árboles, la estimación de abundancia de fitoplancton, la evaluación de diversidad en aves, peces y mamíferos, el mapeo de stocks de carbono, el monitoreo del tráfico ilegal de especies y la calidad del agua, y el impacto de las pesquerías sobre el medio ambiente, entre otros [70].

- Deep Learning en la biomedicina: en la actualidad, la biomedicina está fuertemente enlazada con altos volúmenes de datos relacionados con imágenes médicas, secuencias de genes, estructuras de proteínas y electroencefalogramas, entre otros. El Deep Learning, en especial el desarrollo de modelos de redes neuronales convolucionales ha permitido automatizar la clasificación, estimación y análisis de estos datos, facilitando la comprensión de la salud y las enfermedades del ser humano [71].

- Deep Learning en la química: en la industria química, el Deep Learning ha permitido abordar una gran variedad de problemas químicos, como la mejora de la química computacional, el diseño y optimización de fármacos y materiales químicos, la estimación de estructuras moleculares y reacciones químicas, y la planificación de síntesis [72].

5.3.1 Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (CNN) son un tipo de Redes Neuronales Artificiales que imitan las neuronas de la corteza visual primaria del ojo humano con el objetivo de identificar en las entradas diferentes características de imágenes. Pueden tomar imágenes de entrada y asignar importancia o pesos a sus diferentes elementos para así diferenciar unos de otros. Contienen varias capas ocultas (*hidden layers*) con jerarquías; las primeras capas pueden identificar líneas horizontales, verticales y curvas, las capas más profundas pueden identificar características más complejas, como rostros y siluetas [67] [68].

Una convolución en una red neuronal es una operación lineal, donde se multiplican pesos con los datos de entrada. Como este método fue diseñado para usar entradas en dos dimensiones, la multiplicación se lleva a cabo entre una matriz con los datos de entrada y una matriz bidimensional de pesos, llamada filtro o kernel, la cual es más pequeña que la de entrada, permitiendo que el mismo filtro se multiplique con la matriz de entrada varias veces en distintos puntos de la entrada [75]. En la *Figura 7* se muestra un ejemplo de una convolución, en la cual

se opera la imagen de entrada con el filtro, para obtener una imagen de salida con convolución: de la matriz 7*7 se opera con el filtro una fracción del tamaño 3*3, este resultado es almacenado en la imagen de salida. Después de este paso se repite el proceso mientras el filtro se desplaza hacia la derecha y hacia abajo, para terminar la totalidad de la imagen.

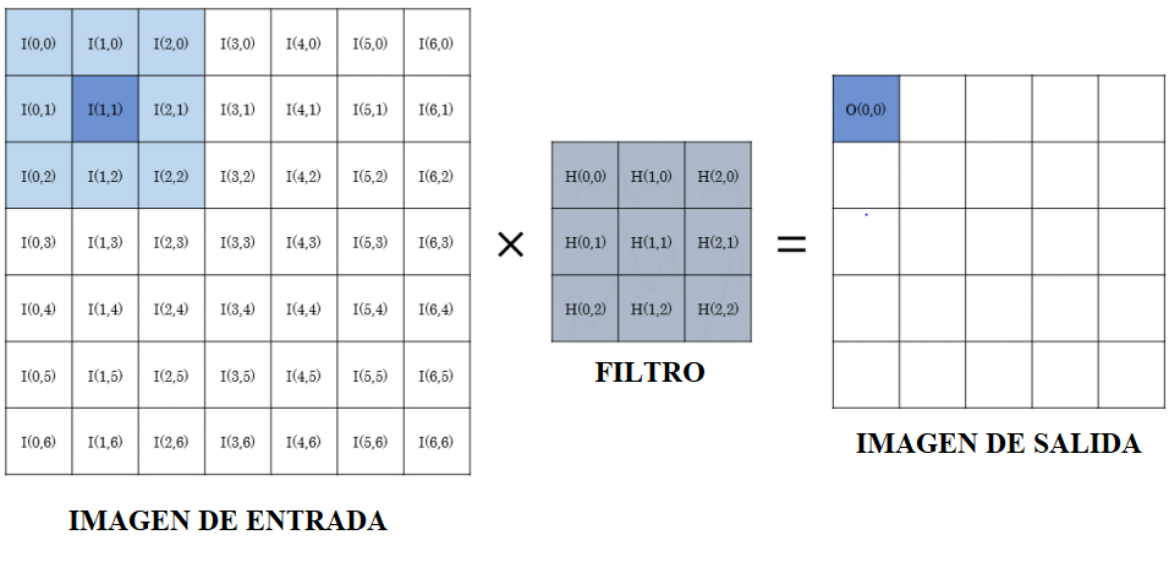

Figura 7. Convolución en una imagen de entrada de 7x7 y un filtro de 3x3. Adaptado de: [76] (traducido).

La capa de entrada de la red son los pixeles de una imagen; en el caso de una imagen de 512x512 pixeles en escala de grises, habrán 262,144 neuronas en la capa de entrada; si la imagen es a color con el esquema de color RGB (3 canales), habrán 786,432 neuronas en dicha capa.

En la etapa de preprocesamiento se debe alimentar la red con valores normalizados entre 0 y 1, dividiendo el valor de los colores de los pixeles que van de 0 a 255, entre 255.

En la siguiente etapa se hacen las convoluciones, tomando grupos pequeños de pixeles de la imagen de entrada y haciendo operaciones con matrices (kernel), para generar matrices de salida, que serán las capas de neuronas ocultas. El kernel opera con la matriz de entrada de izquierda a derecha y de arriba hacia abajo, como se observa en la *Figura 8*. Un conjunto de kernels se llama filtro y la cantidad de filtros indica la cantidad de capas ocultas, por lo tanto, si se operara con 20 filtros, se obtendrían 20 capas ocultas. Este conjunto se denomina *feature mapping*. A medida que se generan las convoluciones, se aplica una capa como función de activación, usualmente no lineal para mayor adaptabilidad del modelo, que permite hacer una estimación en un rango de [0,1] o [-1,1].

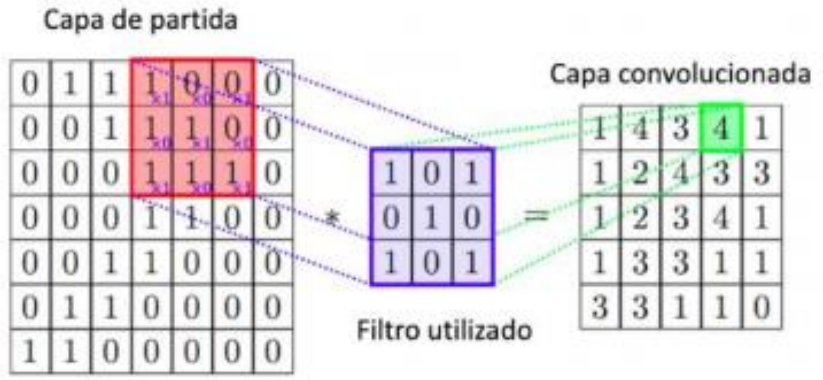

Figura 8. Operaciones del filtro con la capa de entrada para obtener la convolución. Adaptado de: [77]

En el submuestreo se reducen la cantidad de neuronas antes de hacer otra convolución, disminuyendo el alto y el ancho y conservando las principales características que se detectaron en los filtros [78]. Finalmente, se realiza una conexión entre la última capa de submuestreo y una red neuronal tradicional con la función de activación Softmax (ver *Ecuación 4*) y se obtienen como resultado una probabilidad (entre 0 y 1) de que ocurra un suceso dentro del marco de la clasificación [79]. Por ejemplo, si se quiere saber si las imágenes tienen esteganografía [1,0] o no tienen esteganografía [0,1] y se obtiene como resultado [0.20, 0.80], la probabilidad de que la imagen tenga esteganografía es de 20% y la probabilidad de que no tenga es de 80%.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$ (Ecuación 4. Función Softmax. Fuente: [80])

donde,

σ = SoftMax
$(\vec{z})$ = Vector de entrada
e^{z_i} = Función exponencial estándar para el vector de entrada
K = Número de clases en el clasificador multiclase
e^{z_j} = Función exponencial estándar para el vector de salida

En la *Figura 9*, se muestra la arquitectura de las Redes Neuronales Convolucionales.

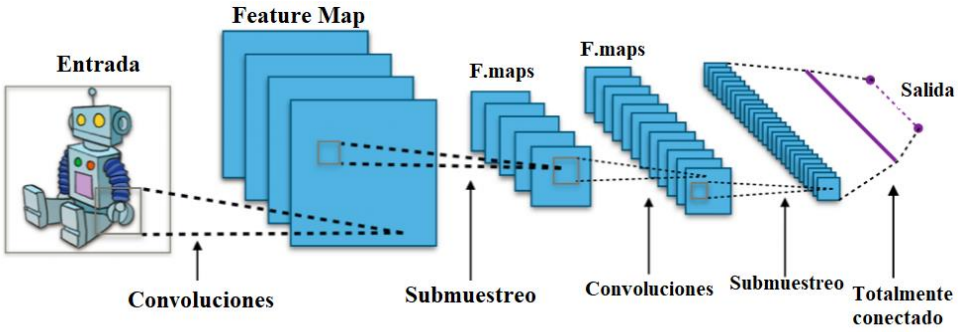

Figura 9. Arquitectura de las Redes Neuronales Convolucionales. Adaptado de: [81] (traducido)

5.4 Metodología de Desarrollo de Software Scrum

La metodología Scrum se caracteriza por procesos iterativos con cambios constantes que tienen un alto grado de incertidumbre. El proceso se lleva a cabo en iteraciones cortas de un tiempo definido, generalmente, de dos semanas, de la siguiente forma:

1. Selección de requisitos: inicialmente se seleccionan los principales requisitos del proyecto que se podrán completar en cada iteración y se establece la meta de la iteración.

2. Planificación de la iteración: se realiza la planificación de la iteración, donde se elabora una lista de tareas necesarias para cumplir con los requisitos del proyecto (Sprint Backlog). Se determina el procedimiento que va a permitir obtener el resultado esperado con el menor esfuerzo.

3. Ejecución de la iteración (Sprint): se inspecciona el trabajo realizado en la iteración, se analiza el estado de la lista de tareas y se evalúan posibles mejoras. Se debe tener en cuenta que, si se reduce la cantidad de requisitos a trabajar en cada iteración, se pueden obtener mejores resultados.

4. Revisión: se realizan adaptaciones en función de los resultados obtenidos en la iteración.

5. Retrospectiva: se identifican los inconvenientes que impidieron que el proyecto se desarrollara de la mejor manera y se determinan formas de mejorar la productividad.

6 METODOLOGÍA

La metodología seguida para el desarrollo del proyecto tiene contempladas las etapas de ciclo de trabajo SCRUM, instrumentos y herramientas, preprocesamiento y modelado.

6.1 Ciclo de trabajo SCRUM

Durante el desarrollo de la investigación se mantuvo una activa documentación de los sprints y los prototipos, relacionándolos con un tablero Kanban de Trello. Mediante la planeación de las iteraciones en los sprint se pudo mitigar los imprevistos en el ciclo de vida del proyecto, además ayudó en el proceso de mejora continua propia de una metodología de prototipado. En el tablero de seguimiento se documentaron las reuniones extraordinarias y se mantuvo a disposición el 100% del material utilizado en el desarrollo del presente proyecto, pudiendo así realizar correcciones asíncronas del proyecto. Fue vital empalmar los avances asíncronos en los correspondientes sprints, de esta forma se optimizó el tiempo de reunión entre investigadores. Estas evidencias se pueden encontrar en el *Anexo 1*.

Después de cada reunión semanal en equipo, se plasmaron los comentarios y tareas teniendo en cuenta el tablero Kanban de Trello, obteniendo al final una lista de tareas por realizar, tareas en desarrollo, temas por investigar y tareas realizadas, como se observa en la *Figura 10*. Los sprints colaborativos permitieron mantener la motivación en el trabajo, mitigar errores en el resultado final, priorizar tareas, reforzar los conocimientos adquiridos a lo largo del programa académico, corregir errores de base y tener siempre presentes los objetivos del desarrollo del proyecto.

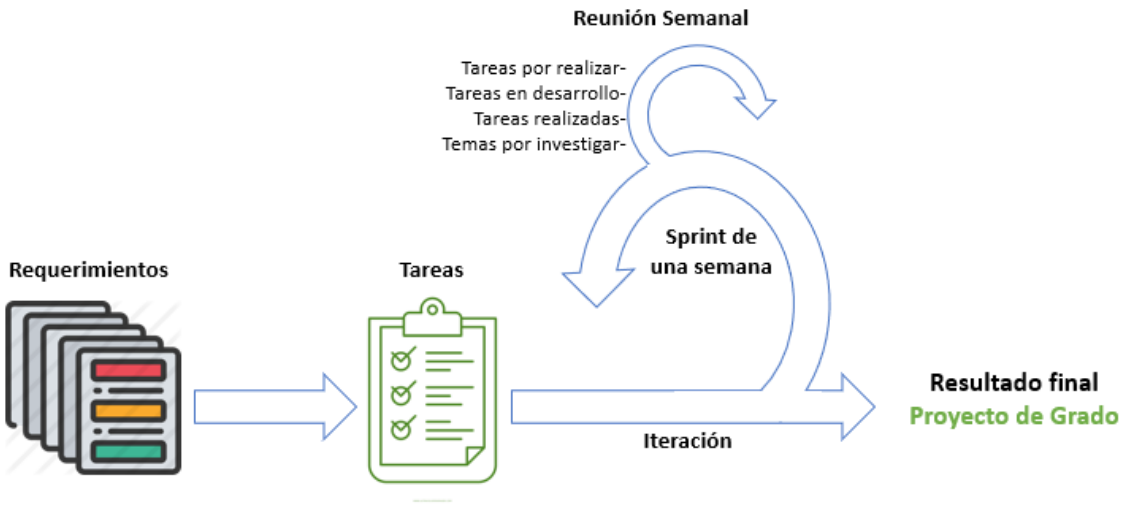

Figura 10. Ciclo de trabajo SCRUM. Fuente: Autor

6.2 Instrumentos y Herramientas

En este apartado se describen los entornos de desarrollo utilizados para llevar a cabo el proyecto, incluyendo componentes de hardware y software.

Este proyecto fue desarrollado utilizando los VPS (Virtual Private Server) suministrados por el CCA (Centro de Computación Avanzada) de la universidad, a los cuales se accedió mediante la herramienta de escritorios remotos Anydesk y el protocolo SSH (Secure Shell). Los recursos otorgados por el CCA fueron los siguientes:

- VPS Stegian_1 (Ubuntu Desktop 16.04 LTS x64 bits): Intel® Xeon® CPU E5-4669 v4 @ 2.20GHz × 8 cores, 12 GB RAM, 530 GB de almacenamiento.

- VPS Stegian_2 (Windows Server 2019 Standard 10.0 x64 bits): Intel® Xeon® Gold 6130 CPU @ 2.10GHz x 32 cores, 48GB RAM, 100 GB de almacenamiento.

Para realizar el procesamiento inicial de los datos se hizo uso de Matlab, plataforma para programación y cálculos numéricos, licenciada por la Universidad Pontificia Bolivariana.

Para el modelado y preprocesamiento se usó el lenguaje de programación Python, ya que se considera el lenguaje estándar para operaciones de Deep Learning y procesamiento de datos, además de permitir un rápido prototipado [82]. El uso de Python se llevó a cabo mediante el gestor de paquetes y ambientes de programación Anaconda, utilizando como IDE (Integrated Development Environment o Entorno de Desarrollo Integrado) JupyterLab, Jupyter Notebook y Spyder. Como librerías de Python, se usaron Tensorflow, Keras, OpenCv, NumPy, Pyplot y Pandas. Como módulos adicionales, se usaron módulos propios para calcular atributos.

Para el control de versiones y cambios se utilizó GitHub, una famosa implementación de Git, para facilitar el acceso a archivos en los diferentes entornos de ejecución. El repositorio utilizado se encuentra privado debido a la naturaleza de esta investigación; sin embargo, debido a la necesidad de realizar revisión de los datos, una copia de los resultados fue agregada al *Anexo 2.*

Las versiones de las diferentes dependencias y herramientas tecnológicas son mencionadas a continuación:

- Matlab R2020b
- Python 3.8.11
- Anaconda 3 2021.05
- JupyterLab 3.1.7
- Jupyter Notebook 6.4.2
- Keras 2.4.3
- OpenCv 4.0.1
- Spyder 5.0.5
- Tensorflow 2.3
- Matplotlib 3.4.2
- Pandas 1.3.1
- Numpy 1.20.3

En caso de requerir acceso al repositorio privado de GitHub, este se otorgará bajo la ley 1581 del 2012 y la ley 1915 de 2018.

6.3 Adquisición de datos

La primera etapa, correspondiente a la adquisición de datos, inició con una búsqueda de conjuntos de datos que incluyeran homogeneidad de clases y una cantidad de elementos mayor a la cantidad de pixeles en una imagen de 512*512, esto con el fin de tener mayor cantidad de observaciones que de características por imagen. Cabe destacar que en esta etapa solo se tuvieron en cuenta conjuntos de datos gratuitos y abiertos para investigación.

En la selección del conjunto de datos, teniendo en cuenta la necesidad de trabajar con elementos heterogéneos, se hizo uso de la base de datos gratuita ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012-2017, que contiene 1,000 clases de objetos, 1,281,167 imágenes de entrenamiento, 50,000 imágenes de validación y 100,000 imágenes de prueba, de las cuales solo se pudo utilizar el subconjunto de entrenamiento, ya que era el único que mantenía las imágenes etiquetadas (con el fin de aplicar técnicas de aprendizaje supervisado), dejando una población de 1,281,167.

En la *Tabla 2* se muestran todos los conjuntos de datos con formato de color sRGB y gratuitos contemplados para este estudio, siendo seleccionado ImageNet, gracias a su facilidad de acceso y uso [18] [77]. El uso de este conjunto de datos es solo con motivos académicos, en ningún momento constituye un uso por fuera de los términos y condiciones acordados al acceder a este.

Tabla 2. Descripción de los conjuntos de datos considerados para esta investigación.

Nombre del dataset	Descripción	Cantidad de imágenes	Tamaño de imágenes	Formato de imagen	Link	Estado
Dresden Image Database	Múltiples cámaras con varias escenas. Usada para pruebas de técnicas forenses.	3,500	512 x 512	JPEG	http://forensics.inf.tu-dresden.de/ddimgdb/ (link roto, consultado por última vez el 01/Julio/2021)	Descartado No se puede verificar la heterogeneidad
Cube+ Dataset	Calibración de color. Cubo negro ubicado en la esquina inferior derecha. 1,707 escenas.	10,242 pares	1,037 x 692	JPEG	http://cvil.eecs.yorku.ca/projects/public_html/sRGB_WB_correction/dataset.html	Descartado No homogéneo, resolución elevada, elemento recurrente en imágenes
MIT-Adobe FiveK rendered Dataset	Calibración de color. 5,000 escenas.	29,980	Mixta (400-800) x (400-800)	JPEG	http://cvil.eecs.yorku.ca/projects/public_html/sRGB_WB_correction/dataset.html	Descartado No se puede verificar la heterogeneidad

Nombre del dataset	Descripción	Cantidad de imágenes	Tamaño de imágenes	Formato de imagen	Link	Estado
MIT-Adobe FiveK Dataset	5,000 escenas	29,980	Mixta (400-800) x (400-800)	TIFF16 y DNG	https://data.csail.mit.edu/graphics/fivek/	Descartado Formato RAW, peso
Color Constancy for Multiple Light Sources	Usado para corrección de color y análisis de hiper espectro.	1,437	Aprox. 335 x 255	-	https://ivi.fnwi.uva.nl/isis/publications/bibtexbrowser.php?key=GijsenijTIP2012&bib=all.bib	Descartado Tamaño
Berkeley Segmentation Data Set and Benchmarks 500	Usada para detección de contornos	500	481 x 321	-	https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html	Descartado Tamaño
Kodak Image Dataset	-	25	768 x 512	-	http://www.cs.albany.edu/~xypan/research/snr/Kodak.html	Descartado Tamaño
Adobe WB dataset	Usado para corrección de color.	65,416 pares	Mixta (800-1,700) x (800-1,700)	-	http://cvil.eecs.yorku.ca/projects/public_html/sRGB_WB_correction/index.html	Descartado Resolución elevada
Hyperspectral Images of Natural Scenes	Análisis de colores por fuera del espectro visible.	30	1,344 x 1,024	-	https://personalpages.manchester.ac.uk/staff/david.foster/Hyperspectral_images_of_natural_scenes_02.html	Descartado Resolución, tamaño

Nombre del dataset	Descripción	Cantidad de imágenes	Tamaño de imágenes	Formato de imagen	Link	Estado
LabelMe – MIT	Proyecto abierto con 658K+ etiquetas para etiquetado de imágenes	190K+	-	JPEG	http://labelme2.csail.mit.edu/Release3.0/	Descartado Dificultad para acceder
Imagenet	Proyecto abierto 100K+ clases divididas en subclases	14M+	-	-	http://www.image-net.org/	Aprobada Cantidad de clases presentes

Fuente: Autor.

Los conjuntos de datos utilizados para el desarrollo del proyecto son descritos a continuación, utilizando combinaciones entre los algoritmos esteganográficos escogidos y los diferentes subconjuntos seleccionados.

Los algoritmos esteganográficos escogidos fueron Miranda LSB, WOW (con payload 0.3) y S-UNIWARD (con payload 0.3). Para los anteriores, se estructuraron diferentes conjuntos de datos utilizando las imágenes en plano, extracción de atributos Miranda y extracción de atributos García. El dataset de Miranda está disponible en [30].

6.4 Preprocesamiento

El preprocesamiento se llevó a cabo en los puntos descritos a continuación, realizando iteraciones sobre algunos apartados para refinamiento, tal como es de esperar en el ciclo de trabajo SCRUM y la metodología prototipada de software.

6.4.1 Limpieza y estructuración de datos

En la limpieza del conjunto de datos, se realizó un estudio de la complejidad espacial de las imágenes para determinar el tamaño adecuado para usar, mediante la herramienta Seaborn de Python, que permite la visualización interactiva de datos. En un principio, se escogió la resolución espacial de 256x256, como se muestra en la *figuras 11-2*. Sin embargo, debidoa limitaciones en la capacidad computacional instalada, finalmente se eligió trabajar con una resolución espacial de 128x128, dejando así una población sobre la que se puede realizar un recorte de 128x128 de 930,000 observaciones, tomando la máxima cantidad de imágenes por categoría y manteniendo la distribución natural de las imágenes. Esto último se realiza para minimizar el sesgo creado por el desbalance de categorías, ya que, de existir, es menos probable que se detecte esteganografía en las clases con menos observaciones.

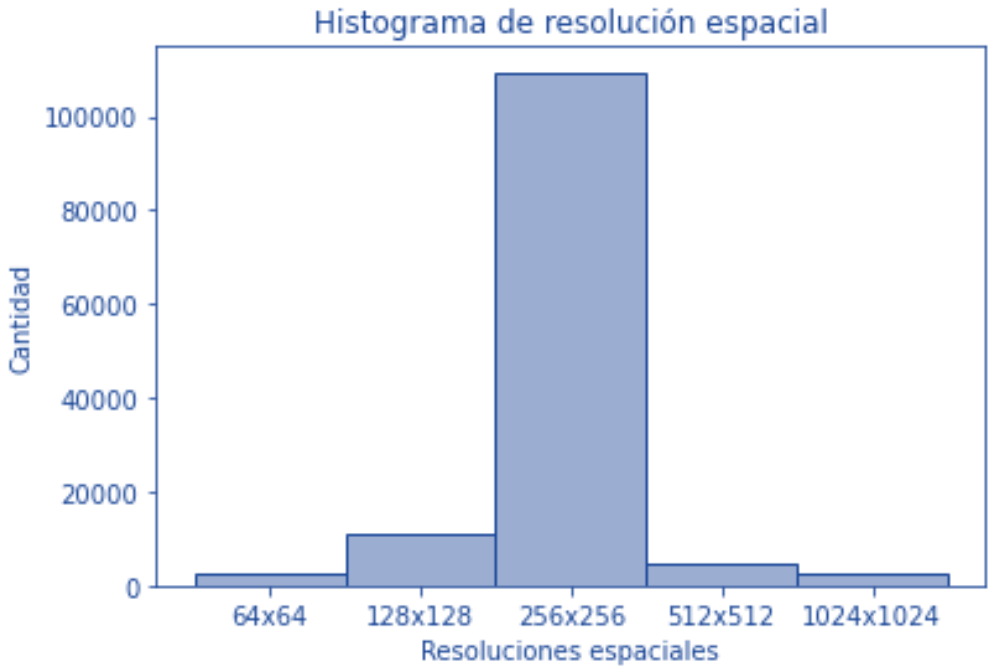

Figura 11. Distribución de la resolución espacial de las imágenes en el conjunto de datos inicial.

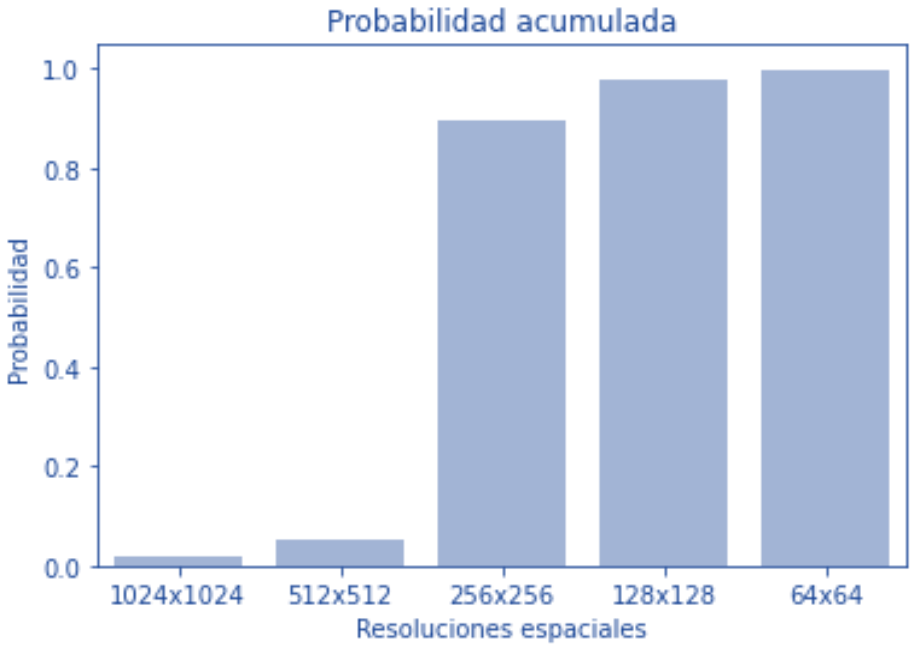

Figura 12. Probabilidad acumulada para la resolución espacial de las imágenes en el conjunto de datos inicial.

En el proceso de limpieza y estructuración de datos se realizó un recorte aleatorio de las imágenes, para establecer las muestras con las resoluciones espaciales seleccionadas. Como medida adicional, se realizó una conversión de imagen del esquema RGB a imágenes monocromáticas implementando el método estándar [84], mencionado en la *ecuación 5*.

$$gray = 0.2990 * R + 0.5870 * G + 0.1140 * B$$ (Ecuación 5. Conversión de imagen RGB a escala de grises. Fuente: [84]).

donde,

$R = canal\ rojo$
$G = canal\ verde$
$B = canal\ azul$

6.4.2 Adición de esteganografía

Se usó Matlab para embeber esteganografía en los archivos digitales de imagen, utilizando los algoritmos proporcionados por Fridrich [85], que permite realizar la comparación entre los resultados obtenidos por otros investigadores. Se desarrolló un algoritmo cuya función era tomar las imágenes organizadas y adicionar la carga esteganográfica de forma aleatoria en uno de los tres canales RGB. En un principio se utilizaron los algoritmos WOW, S-UNIWARD, MiPOD, HUGO, MG y MVG; cada uno con variaciones de 0.1, 0.2, 0.3, 0.4 y 0.5 de payload para todos los casos.

Después de estudiar los algoritmos mencionados, se encontraron diferencias notables entre el payload de entrada y el de salida, por lo que se profundizó en el análisis de estos. En la ecuación sugerida por los autores de los conjuntos de datos para contar el payload de salida, no se contemplan los cambios negativos de intensidad para la adición del payload. Se pudo comprobar la poca asimilación del payload de entrada en todos los algoritmos escogidos, situación que puede ser apreciada en las en las *figuras 13-16*. Se determinó que WOW y S-UNIWARD, siendo los más usados por los referentes en el estado del arte, tienen menor complejidad temporal y mejor asimilación del payload entre los preseleccionados (algunos algoritmos no permiten payloads superiores a 1.0), siendo estos los algoritmos escogidos para continuar con el desarrollo del proyecto.

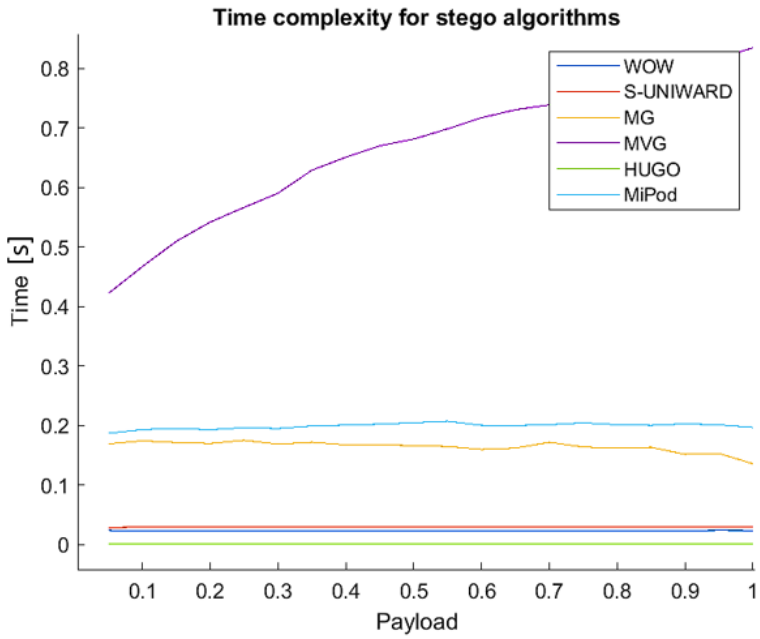

Figura 13. Complejidad temporal para métodos esteganográficos propuestos.

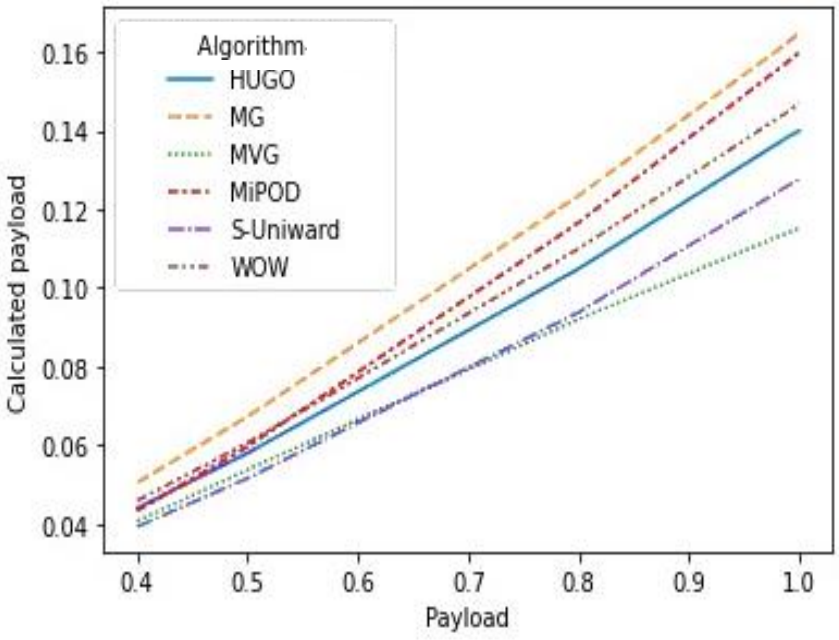

Figura 14. Asimilación de payload para algoritmos propuestos.

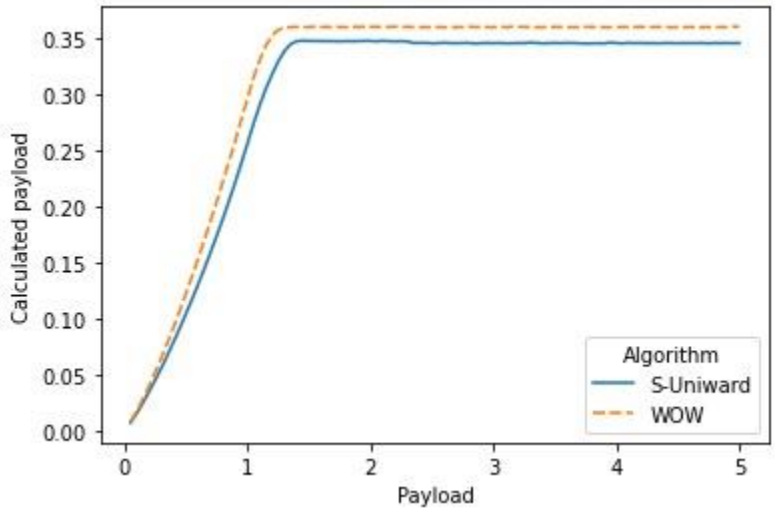

Figura 15. Asimilación de payload para algoritmos acotados.

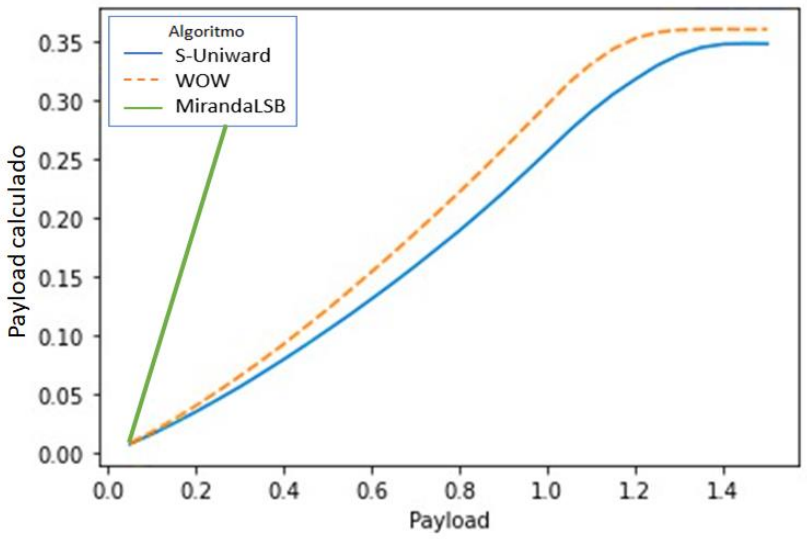

Figura 16. Asimilación detallada de payload para algoritmos acotados.

Adicionalmente, se implementó el método LSB estándar documentado por Miranda [30], de tal forma que un total de tres tipos de algoritmos esteganográficos fueron considerados: WOW, S-UNIWARD y Miranda LSB. En la *tabla 3* se muestra el PSNR (Peak signal-to-noise ratio), que indica el la capacidad de un método esteganográfico para embeber contenido [86].

Método esteganográfico	**PSNR medio**
MirandaLSB	54.08 dB
WOW	10.16 dB
S-UNIWARD	10.90 dB

Tabla 3. PSNR medio para los algoritmos esteganográficos utilizados. Fuente: Autor.

La distribución de carga esteganográfica para los algoritmos escogidos se puede apreciar en las *figuras 17-19*, donde *cover* hace referencia a las imágenes portadoras y *Embedding changes* hace referencia a los pixeles alterados.

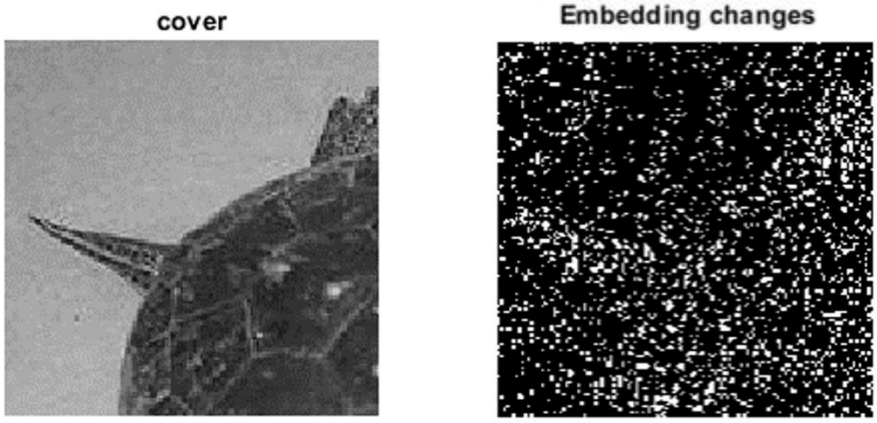

Figura 17. Distribución esteganográfica (a la izquierda) para S-UNIWARD 0.3, a partir de la imagen portadora (a la derecha).

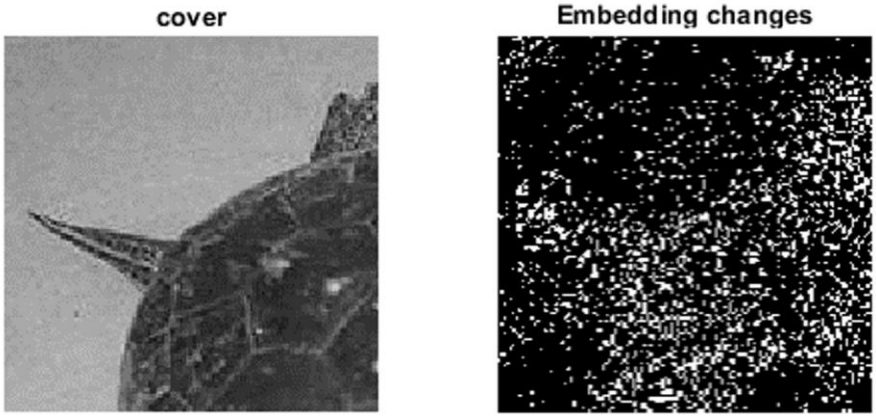

Figura 18. Distribución esteganográfica (a la izquierda) para WOW 0.3, a partir de la imagen portadora (a la derecha).

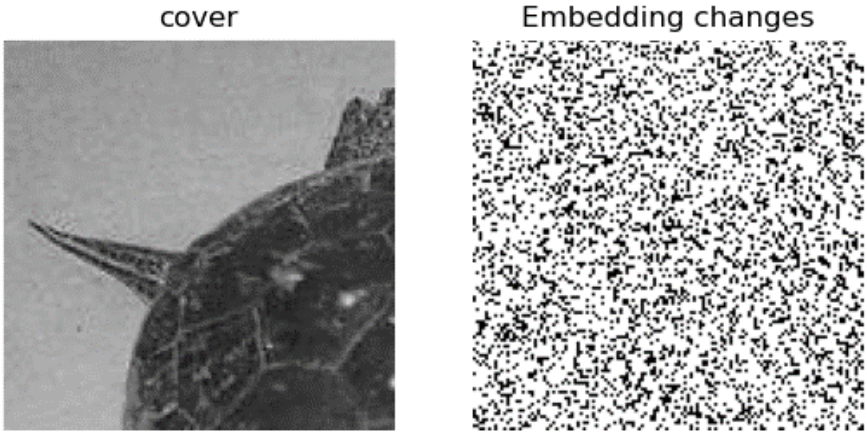

Figura 19. Distribución esteganográfica (a la izquierda) para MirandaLSB 0.5, a partir de la imagen portadora (a la derecha).

6.4.3 Extracción de atributos

Esta etapa suele utilizarse en proyectos de Deep Learning de forma intrínseca a las capas de convolución, ya que el proceso de aprendizaje de atributos es realizado por las redes neuronales en su primera capa; sin embargo, se incluyó en la etapa previa de modelado para incorporar posibles patrones adicionales, siguiendo la investigación de Miranda [29], siendo un enfoque que ha sido usado satisfactoriamente en diversos escenarios como un enfoque funcional [87]–[91]. Se incorporó un procedimiento de extracción de atributos por ventanas de imagen como un enfoque complementario, con el fin de comparar resultados con los encontrados por la aplicación de métodos formales de Deep Learning.

6.5 Modelado

El modelado de las redes neuronales convolucionales se desarrolló por medio de Python, teniendo en cuenta los elementos analizados en el estado del arte. El proceso se llevó a cabo tomando como entrada imágenes RGB de 128x128, así que se obtuvieron 49,152 neuronas de entrada, a las cuales se aplicaron filtros para obtener imágenes de salida iguales a la cantidad de filtros; seguido a esto, se realizó el submuestreo, donde se redujo el tamaño de las imágenes, conservando las características principales de estas y después se aplicó la función de activación Softmax, encargada de dar como resultado final una probabilidad de presencia de esteganografía.

En la sección de resultados se detallan todas las iteraciones del modelado que fueron llevadas a cabo utilizando búsqueda manual de hiperparámetros, validación cruzada por 10-Fold y gradiente descendente estocástico (SGD) para optimizar la función de pérdida.

7 RESULTADOS

En este apartado se documentan los resultados obtenidos en prototipos y experimentos, siendo estos últimos, los métodos utilizados para cotejar esta investigación con el estado del arte y sus diferentes enfoques.

7.1 Desarrollo del algoritmo de Deep Learning

El desarrollo del algoritmo de de detección de esteganografía LSB se hizo a partir de modelos de Redes Neuronales Convolucional (CNNs) propuestos y descritos en la sección anterior utilizando las bibliotecas Keras y TensorFlow. La optimización de hiperparámetros fue realizada por búsqueda manual, para garantizar la heterogeneidad de categorías se usó muestro estratificado y se incluyó la etapa de validación por 10-Fold. A lo largo del desarrollo, se crearon distintos prototipos aquí documentados; sin embargo, para contrastar los resultados obtenidos con el estado del arte y encontrar resultados más satisfactorios según lo esperado en el Estado del Arte, se ejecutaron experimentos adicionales.

Algunos prototipos fallidos (por circunstancias mayormente técnicas ajenas al investigador) están documentados en el *Anexo 3* junto a su causa de fallo.

En cada uno de los prototipos, el orden de documentación de la arquitectura de capas es el siguiente: *Input* (capa de entrada), *Conv2D* (capa con convolución), *MaxPooling2D* (capa de pooling o submuestreo, que reduce la cantidad de parámetros), *Flatten* (capa donde se convierten las matrices multidimensionales en una matriz única), *Dropout* (capa donde se desactivan algunas neuronas de forma aleatoria para reducir la dependencia entre neuronas cercanas), *Dense* (capa densamente conectada) y *Output* (capa de salida). Después de *Conv2d* puede haber una capa llamada *BatchNormalization*, que normaliza y estandariza cada lote de datos para estabilizar la red neuronal convolucional y hacerla más rápida.

En todos los prototipos se aplicó la función lineal ReLU como función de activación en las capas Conv2D y en Dense; esta función se encarga de llevar a cero todos los valores negativos y aceptar los valores positivos. En las capas de Output se aplicó la función Softmax para obtener una salida binaria en la clasificación (con esteganografía o sin esteganografía). La cantidad de neuronas por capa hace referencia a la cantidad de neuronas en cada una de las *n* capas del tipo mencionado en la tabla.

A continuación, se describen los prototipos desarrollados distribuidos en dos secciones: (a) los prototipos aplicados a la detección de esteganografía en imágenes monocromáticas, y (b) los prototipos aplicados a la detección de esteganografía en imágenes a color.

7.1.1. CNN con imágenes monocromáticas

Prototipo 1: como se observa en la tabla 4, en este prototipo, las entradas fueron imágenes monocromáticas de 128*128 parte de los ya mencionados tres conjuntos de datos. Se establecieron 128, 256 y 1024 neuronas por cada capa de tipo Conv2D; 1024, 512, 256, 128, 64, 32, 16, 8 y 4 neuronas por cada capa densa y un output de 2 neuronas. Los resultados obtenidos por este prototipo son: precisión del 50.00% y F1 score de 65.20%.

Tabla 4. Resumen del prototipo 1.

Capas	Neuronas por capa	Tamaño del kernel/dropout	Activación
Input	128	-	-
Conv2D	128, 256 y 1024	5	ReLU
MaxPooling2D	-	2	-
Flatten	-	-	-
Droput	-	0.2	-
Dense	1024, 512, 256, 128, 64, 32, 16, 8 y 4	-	ReLU
Output	2	-	Softmax

Prototipo 2: en la tabla 5 se presenta la arquitectura de este prototipo, que recibe como entrada imágenes monocromáticas de 128*128. Se agregó la capa BatchNormalization, se utilizó Stochastic Gradient Descent (SGD) como optimizador. En las capas Conv2D, a diferencia del prototipo 1, se establecieron 512 y 1021 neuronas por capa, mientras que para capas densas 1024, 512, 256, 128 y 8 neuronas por capa, quedando al final una neurona de salida. Se obtuvieron métricas de precisión del 50.00% y F1 score de 66.70%, siendo mayores que en el prototipo anterior.

Tabla 5. Resumen del prototipo 2.

Capas	Neuronas por capa	Tamaño del kernel/dropout	Activación
Input	128	-	-
Conv2D	512 y 1024	5	ReLU
BatchNormalization	-	-	-
MaxPooling2D	-	3	-
Flatten	-	-	-
Droput	-	0.2	-
Dense	1024, 512, 256, 128 y 8	-	ReLU
Output	1	-	Softmax

Prototipo 3: en la tabla 6, se presenta el prototipo 3, con imágenes monocromáticas de 128*128 y ADAM. Se implementó optimizador ADAM para reducir el tiempo de procesamiento. Se obtuvieron 128, 256, 512 y 1024 neuronas por capa en Conv2D y 1024, 512, 256, 128 y 8 neuronas en la capa Dense, con un Output de 2 neuronas por capa. Las métricas obtenidas fueron

de 66.90% para F1 score y 49.00% de precisión, siendo similares a las obtenidas en el prototipo anterior.

Tabla 6. Resumen del prototipo 3.

Capas	Neuronas por capa	Tamaño del kernel/dropout	Activación
Input	128	-	-
Conv2D	128, 256, 512 y 1024	5	ReLU
MaxPooling2D	-	3	-
Flatten	-	-	-
Droput	-	0.2	-
Dense	1024, 512, 256, 128 y 8	-	ReLU
Output	2	-	Softmax

Prototipo 4: el prototipo 4, presentado en la tabla 7, se trabajó con enfoque de atributos por ventanas de 16*16. Este prototipo recibe imágenes a monocromáticas de 128*128 y les realiza extracción de atributos por ventanas de 16*16, esto es pasado a la CNN documentada en la siguiente tabla. Se definieron 256, 512 y 1024 neuronas por capa en Conv2D y 1024, 512, 256, 128 y 8 por capas densas, con una salida de una neurona. Las métricas obtenidas fueron: F1 score de 70.80% y precisión del 55.70%, lo que resulta en un desempeño superior a los prototipos anteriores La arquitectura de este prototipo de detección de esteganografía en imágenes monocromáticas es la base para los prototipos siguientes de detección sobre imágenes a color.

Tabla 7. Resumen del prototipo 4.

Capas	Neuronas por capa	Tamaño del kernel/dropout	Activación
Input	256	-	-
Conv2D	256, 512 y 1024	5, 3 y 3	ReLU
MaxPooling2D	-	3	-
Flatten	-	-	-
Droput	-	0.2	-
Dense	1024, 512, 256, 128 y 8	-	ReLU
Output	1	-	Softmax

7.1.2. CNN con imágenes a color

Prototipo 5: este prototipo de CNN, plasmado en la tabla 8, fue propuesto para la entrada de un solo canal de las imágenes RGB 128*128. Se agregó capa de Dropout con el fin de disminuir la carga computacional y facilitar la convergencia. Se utilizó SGD como optimizador. Se obtuvieron 218, 256 y 1024 neuronas por capa en Conv2D y 1024, 512, 256, 128, 8 y 4 en Dense. Se propuso un Output de 2 neuronas; sin embargo, no se pudo encontrar punto de convergencia en el prototipo, debido a que el proceso de embeber esteganografía se realizó aleatoriamente por canal, por lo cual las muestras marcadas como esteganografía ocasionaron excesivos falsos positivos.

Tabla 8. Resumen del prototipo 5.

Capas	Neuronas por capa	Tamaño del kernel/dropout	Activación
Input	-	-	-
Conv2D	128, 256 y 1024	5	ReLU
MaxPooling2D	-	3	-
Flatten	-	-	-
Droput	-	0.2	-
Dense	1024, 512, 256, 128, 8 y 4	-	ReLU
Output	2	-	Softmax

Prototipo 6: En la tabla 9, se presenta este prototipo, que recibe como entrada imágenes a color de 128*128. Se utilizó SGD como optimizador. En la capa Conv2D, se propusieron 128, 256 y 256 neuronas por capa y en la capa Dense256 neuronas. Los resultados obtenidos por este prototipo son: precisión del 50.00% y F1 score de 66.00% con un Output de 10 neuronas.

Tabla 9. Resumen del prototipo 6.

Capas	Neuronas por capa	Tamaño del kernel	Activación
Input	128	-	-
Conv2D	128, 256 y 256	5, 3 y 3	ReLU
MaxPooling2D	-	2	-
Flatten	-	-	-
Dense	256	-	ReLU
Output	10	-	Softmax

Como resultado, el algoritmo para la detección de esteganografía en imágenes digitales a color obtuvo métricas de desempeño de 50.00% de precisión y 60.00% de F1 score consistentes con la investigación de Mamada [24], secundando así sus conclusiones y sentando la hipótesis de que se deben utilizar formatos sin compresión para poder diferenciar más fácilmente los cambios embebidos y obtener un mejor resultado.

En la tabla 10 se relaciona el desempeño de los prototipos propuestos. Puede notarse que el prototipo con mejor desempeño es el cuatro, con un desempeño que supera el documentado en algunos apartados de investigaciones referentes. Este prototipo fue desarrollado para imágenes monocromáticas y sentó las bases de la detección de esteganografía a color (prototipo funcional seis).

Tabla 10. F1 score por prototipo.

Prototipo	Tipo de imagen	F1 Score
1	Monocromática	65.20%
2	Monocromática	66.70%
3	Monocromática	66.90%
4	Monocromática	70.80%
5	A color	No converge
6	A color	66.00%

En la tabla 11 se muestra el desempeño del algoritmo de detección descrito en el prototipo seis sobre imágenes de prueba con contenido esteganográfico derivado de los tres algoritmos considerados. Puede observarse que el desempeño es uniforme entre los métodos esteganográficos, aun cuando para WOW y S-UNIWARD se obtiene un desempeño ligeramente más elevado. Esto puede deberse a la concentración de esteganografía que se incluye en las imágenes digitales sobre algunas zonas de la imagen, lo que facilita su detección.

Tabla 11. F1 score por método esteganográfico.

Algoritmo	F1
MirandaLSB	66.70%
WOW	66.90%
S_UNIWARD	66.90%

El desempeño del conjunto de datos Miranda (precisión del 50.00% y F1 score del 66.40%) se debe a las pocas observaciones del conjunto de datos, considerando que el número de atributos es mayor al número de observaciones, lo que puede causar una solución de un problema sub-dimensionado.

En la *figura 20*, se presenta la complejidad temporal del algoritmo implementado para la detección de esteganografía sobre imágenes a color, donde se observa que la complejidad temporal en prueba (después de haber sido entrenado) es lineal. De la figura anterior, puede observarse la capacidad del algoritmo para procesar 1,000 imágenes en menos de un minuto, siendo este un aporte para coadyuvar en la solución de la ya mencionada dificultad de procesar grandes volúmenes de imágenes en poco tiempo, y superando el tiempo que le tomaría a una persona realizar este procedimiento manualmente.

El modelo de CNN que compone el algoritmo de detección de esteganografía mencionado tiene un tiempo medio de estimación sobre el conjunto de prueba de 55.95 Milisegundos, mientras que el tiempo de entrenamiento es de aproximadamente 75 minutos por epoch, con un total de 20 epochs ejecutados con batch size de 64. Se ejecuta esta cantidad de epochs debido a que es el nivel de tolerancia establecido por el método EarlyStopping (método que sirve para controlar la cantidad de epochs del algoritmo después de alcanzar su punto de convergencia sobre un mínimo usualmente local).

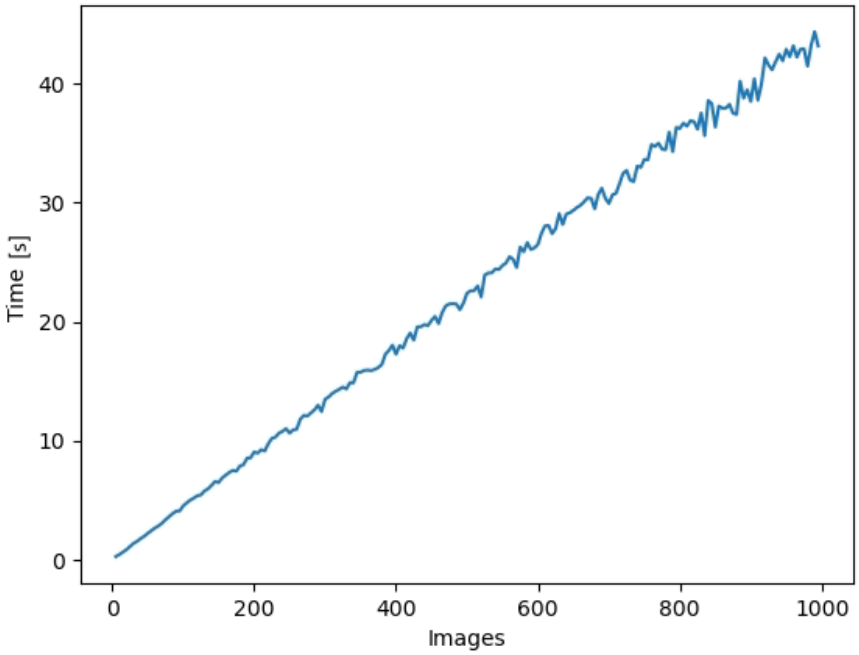

Figura 20. Complejidad temporal del algoritmo implementado.

Además de los prototipos ya documentados, se realizaron variaciones intercalando las funciones de activación y se modificaron los parámetros estándar para los métodos de optimización, prototipos que no se encuentran documentados ya que obtuvieron un resultado similar al prototipo uno.

Adicionalmente a lo propuesto, se exploraron otros tipos de modelos más interpretables y menos caja negra, a los que se les llamará experimentos y resultados adicionales.

7.2 Experimentos y resultados adicionales

Teniendo en cuenta el estado del arte, el primer paso a implementar fue aplicar la extracción de atributos, así contrastar los resultados obtenidos por los prototipos contra nuevos enfoques. Se propusieron atributos adicionales a los de la investigación de Miranda [29], dentro de los que se encuentran: Garcia GeoMean y Epsilon GeoMean.

Según se puede ver en la ecuación 6, el valor de la media geométrica resultará cero en caso de encontrar un dato en ceros. Cabe aclarar que estas implementaciones de la media geométrica se realizaron con el histograma de intensidades por imagen en todos los casos.

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \dots x_n}$$ (Ecuación 6. Media Geométrica. Fuente: Manual de fórmulas y tablas matemáticas, Murray- Spiegel).

Se modificaron los atributos por extraer con el fin de evitar la obtención de atributos con valor cero, ya que esto podría perjudicar el aprendizaje de los modelos que componen el algoritmo de

detección. Se reemplazó la media geométrica documentada por Miranda con los dos siguientes atributos:

- Garcia GeoMean: implementación de la media geométrica estándar sobre el histograma de intensidades por imagen, descartando los datos con valor cero, como se puede ver en la *ecuación 7*.

$$Garcia\ GeoMean = (\prod_{i=1}^{n} \mathbb{1}[x_i \neq 0])^{\frac{1}{n}}$$ (Ecuación 7. García GeoMean. Fuente: Autor).

donde,

$\mathbb{1}[x_i \neq 0] =$ *valores del histograma, omitiendo aquellos elementos con valor en cero*

- Epsilon GeoMean: se agrega épsilon =1 a todos los datos del histograma, para evitar tener datos con valor cero y así modificar la media geométrica resultante.

$$Epsilon\ GeoMean = (\prod_{i=1}^{n} (x_i + 1))^{\frac{1}{n}}$$ (Ecuación 8. Epsilon GeoMean. Fuente: Autor).

Este proceso fue realizado sobre el conjunto de datos en general, no con un enfoque de filtro personalizado para capas convolucionales modificadas.

Se realizó contraste con regresión logística, ANN y CNN, utilizando los conjuntos de datos como se muestra en la *tabla 12.*

Tabla 12. Utilización de los conjuntos de datos.

	Regresión logística	**ANN**	**CNN**
Dataset Miranda [30]	X	X	X
Subset de 5,000 imágenes con MirandaLSB	-	-	X
Atributos Miranda con subset Miranda	X	X	-
Atributos García con subset Miranda	X	X	X
WOW con payload 0.3	-	-	X
Atributos Miranda de WOW 0.3	X	X	
Atributos García de WOW 0.3	X	X	X
S-UNIWARD con payload 0.3	-	-	X
Atributos Miranda de S-UNIWARD 0.3	X	X	-
Atributos García de S-UNIWARD 0.3	X	X	X
MirandaLSB con payload de 0.5	-	-	X
Atributos Miranda de MirandaLSB 0.5	X	X	-
Atributos García de MirandaLSB 0.5	X	X	X

A continuación, en la *figura 21*, se presenta la relación lineal (correlación) entre los atributos, siendo el índice que tiende a rojo el que representa una correlación positiva (proporcionalidad directa), y el que tiende a azul, el de correlación negativa (proporcionalidad inversa); la intensidad del color denota la magnitud de correlación en valor absoluto. En la *figura 21*, tres matrices de correlación son presentadas para: (a) los atributos en el Dataset Miranda, (b) la implementación García de atributos Miranda sobre conjunto de datos Miranda, y (c) los atributos García sobre el conjunto de datos Miranda.

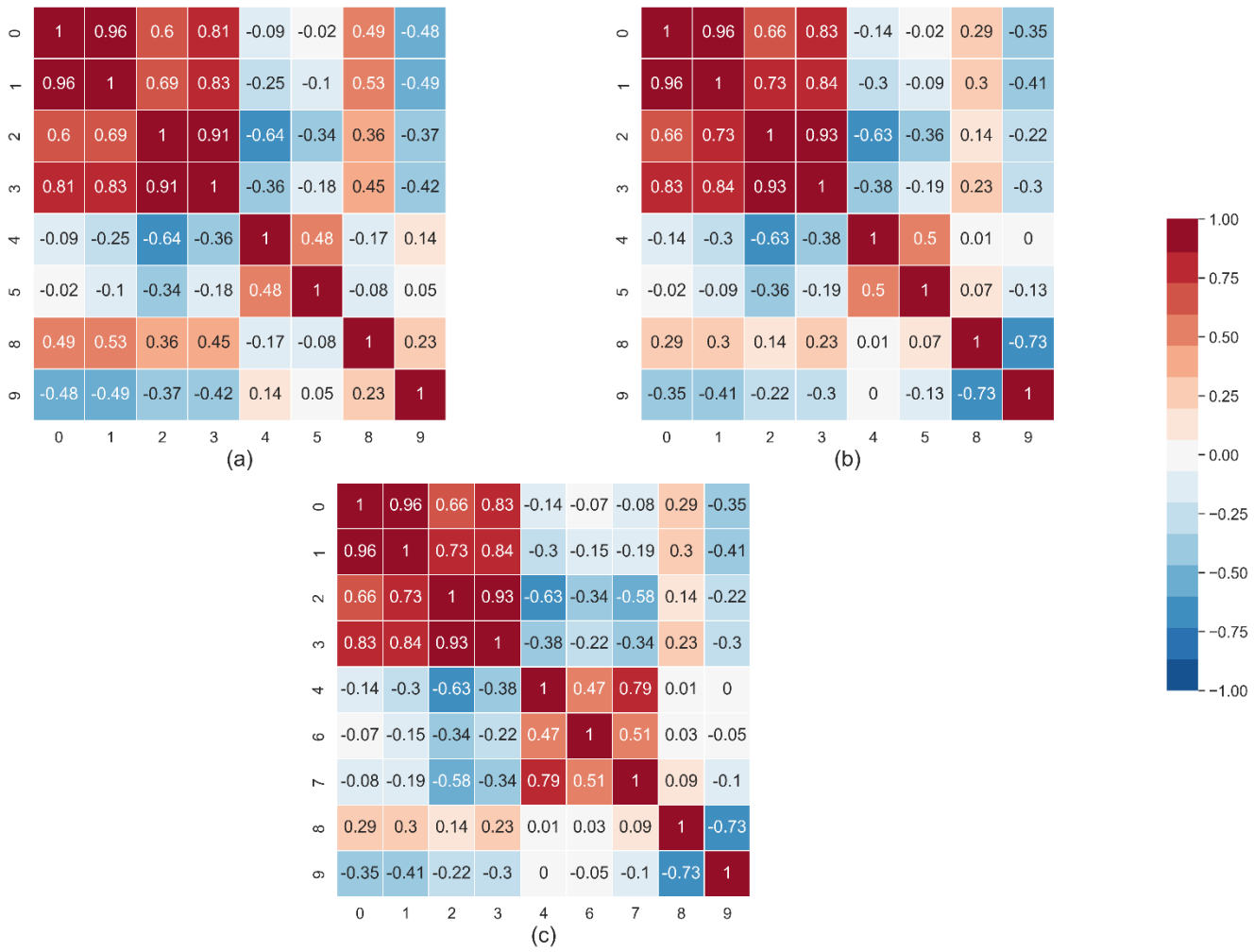

Figura 21. Matrices de correlación entre atributos: (a) Matriz de correlación utilizando el Dataset Miranda [30], (b) Matriz de correlación para implementación García de atributos Miranda sobre dataset Miranda y (c) Matriz de correlación para atributos García sobre dataset Miranda.

A pesar de que el dataset de atributos Miranda [30] tiene 70,000 observaciones extraídas usando Matlab, al emplear un subconjunto de 5,000 observaciones y la extracción de atributos Miranda en Python, los resultados son similares en cuanto a la correlación de atributos. Lo anterior indica

que la extracción de atributos realizada en ese proyecto es consistente con la propuesta por Miranda [29], validando así el estudio de atributos planteado en los experimentos.

7.2.1. Experimentos con Redes Neuronales Artificiales (ANN)

Las redes neuronales convolucionales guardan relación con las redes neuronales artificiales, siendo las primeras una arquitectura de las segundas. En ambos modelos se realiza la implementación de una serie de capas densamente conectadas. La principal diferencia es que, mientras en las CNN se utilizan capas de convolución y *pooling* (normalizado de la imagen) para la extracción de atributos, este es un proceso que se debe realizar previo a la entrada a las ANN.

Para todos los experimentos con ANN se utilizó ADAM como optimizador. En las *tablas 12 - 15* se muestra la arquitectura de diversos experimentos de ANN con enfoques de atributos utilizando conjuntos de datos distintos y proponiendo arquitecturas variables.

1. **Experimento 1:** experimento de ANN con enfoque de atributos García. Se obtuvieron métricas de precisión del 49.40% y F1 score del 66.10%

Tabla 13. Resumen de parámetros del experimento 1.

Capas	Neuronas por capa	Activación
Input	9	-
Dense	8, 4, 2 y 1	ReLU
Output	1	Sigmoid

2. **Experimento 2: e**xperimento de ANN con enfoque de atributos García. Las métricas obtenidas fueron 50.10% de precisión y 66.80% de F1 score.

Tabla 14. Resumen de parámetros de experimento 2.

Capas	Neuronas por capa	Activación
Input	9	-
Dense	9, 4, 2 y 1	ReLU
Output	1	Sigmoid

3. **Experimento 3: e**xperimento de ANN con enfoque de atributos, utilizando dataset Miranda [30]. Las métricas obtenidas fueron precisión y F1 score de 92.80%.

Tabla 15. Resumen de parámetros del experimento 3.

Capas	Neuronas por capa	Activación
Input	8	-
Dense	8, 4, 2 y 1	ReLU
Output	1	Sigmoid

4. **Experimento 4: e**xperimento de ANN con enfoque de atributos, utilizando subset Miranda de 5,000 imágenes y extracción de atributos García. Se obtuvieron métricas de 92.80% para precisión y 93.00% para F1 score.

Tabla 16. Resumen de parámetros del experimento 4.

Capas	Neuronas por capa	Activación
Input	9	-
Dense	9, 4, 2 y 1	ReLU
Output	1	Sigmoid

7.2.1. Experimentos con regresión logística

La regresión logística es uno de los modelos más interpretables, razón por la cual se entrenaron diferentes modelos para cada experimento. Además, se usaron métodos para medir la importancia de cada atributo en el resultado de la estimación.

Como hiperparámetros para los siguientes experimentos, están documentados: (a) el valor para establecer límites en la capacidad de aprendizaje y evitar el overfitting (hiperparámetro *C*), (b) la cantidad máxima de iteraciones para entrenar (hiperparámetro *max_iter*), (c) el tipo de penalización utilizado para regular los pesos de cada atributo (hiperparámetro *penalty*), y (d) la tolerancia del modelo, que es similar a la tolerancia establecida en redes neuronales para la optimización de la función de pérdida (hiperparámetro *tol*).

En la *tabla 16* se muestran el valor de los hiperparámetros definidos para el modelo de regresión logística.

Tabla 17. Resumen de los hiperparámetros especificados para los modelos de regresión logística.

Hiperparámetro	Valor
C	1
max_iter	10
penalty	l2
tol	0.0001

5. **Experimento 5: e**xperimento de regresión logística con enfoque de atributos sobre dataset Miranda [30]. Se obtuvo una precisión del 90.00% y F1 score de 90.00%.

6. **Experimento 6:** experimento de regresión logística con datasets establecidos, aplicando extracción de atributos Miranda y García. Se obtuvieron métricas de Precisión entre 51.00%-52.00% y F1 score de 48.00% -48.50%.

7. **Experimento 7:** experimento de regresión logística con subconjunto del dataset Miranda de 5,000 imágenes para comparar extracción de atributos Miranda y extracción de atributos García. Las métricas obtenidas fueron F1 score de 85.00% y precisión del 86.00%.

Como se puede evidenciar en la *figura 22*, hay cierta diferencia de pesos entre los distintos conjuntos de atributos documentados, donde, a mayor magnitud del peso referenciado, más importancia tiene el atributo en la detección de una imagen.

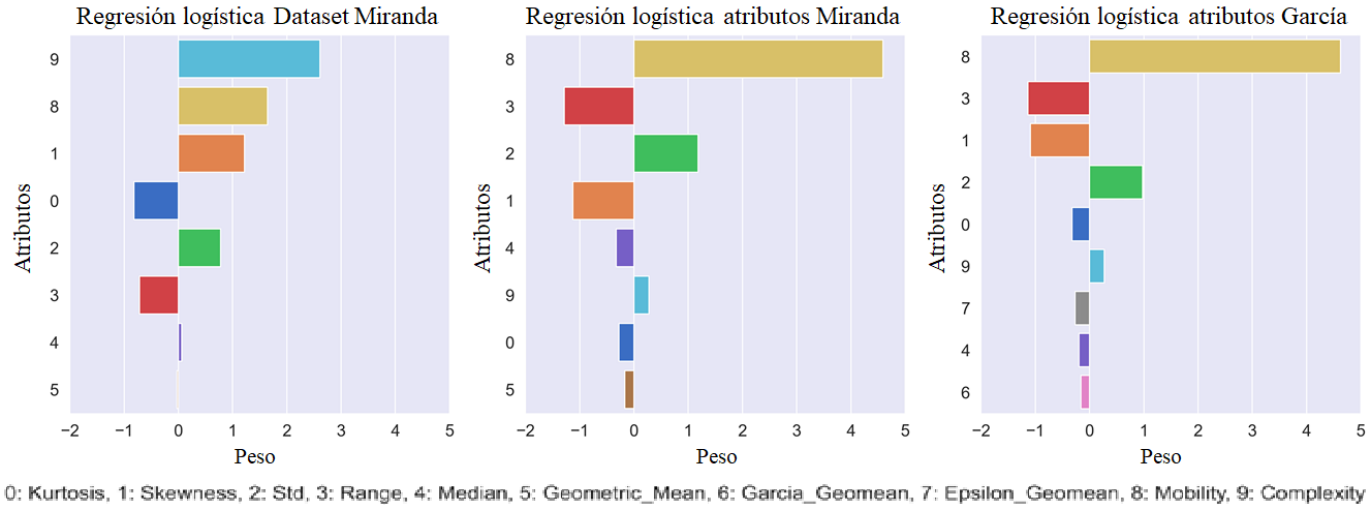

Figura 22. Pesos por atributo para: (a) Dataset Miranda, (b) subconjunto del dataset Miranda utilizando atributos Miranda y (c) subconjunto del dataset Miranda usando atributos García.

Según las figuras anteriores, el atributo con mayor peso para los conjuntos de datos propuestos es la Movilidad, siendo consistente con lo afirmado por Miranda [29], quien, además de relacionar directamente las alteraciones espaciales en las imágenes con la Movilidad de Hjorth, sostiene que con este atributo se podría calcular la cantidad de payload embebido en cada imagen.

Al evaluar todas las posibles combinaciones de atributos en el entrenamiento, se obtuvo que el subconjunto de atributos con mayor precisión (83.30%) fue: {Skewness, Std, Range, Median, Garcia_Geomean, Epsilon_Geomean, Mobility y Complexity}, obteniendo un 0.3% de mejora ante la precisión del 83.00% de los atributos Miranda.

Contrastando los resultados obtenidos, se puede apreciar una clara diferencia de precisión entre el dataset con imágenes JPEG y BMP, siendo estas últimas en las cuales es más probable detectar esteganografía LSB. El contraste entre los atributos Miranda y Garcia, permite apreciar una mejora en la detección utilizando estos últimos.

Los experimentos aquí plasmados fueron realizados sobre imágenes monocromáticas con el fin de economizar recursos computacionales y con un propósito comparativo, cuyos resultados presentados se encuentran agrupados en la *tabla 18*.

Tabla 18. F1 score por experimento.

Experimento	F1 Score
1	0.661
2	0.668
3	0.928
4	0.930
5	0.920
6	0.480
7	0.855

Según los resultados de los experimentos, se puede observar que se obtuvo un mayor desempeño en comparación con los prototipos propuestos de Deep Learning. Esto resulta contradictorio a la naturaleza del método, ya que a mayor complejidad se esperaría obtener un mejor desempeño. Sin embargo, en el contexto de la detección de esteganografía, esta investigación ha podido comprobar que modelos con menor complejidad y mayor interpretabilidad tienen un desempeño más alto en la detección con imágenes monocromáticas de prueba, pudiendo detectar esteganografía con un 93.00% de F1 score.

8 CONCLUSIONES

El proyecto de grado cumplió con todos los objetivos propuestos, tal que se desarrollaron algoritmos para la detección de esteganografía LSB en imágenes digitales a color, a partir de modelos de aprendizaje profundo (Deep Learning). Además de completar los objetivos, se profundizó en la prueba enfoques adicionales contemplados en el estado del arte para la detección de esteganografía, concluyendo que el nivel de complejidad de los modelos no siempre es proporcional a la obtención de resultados exitosos, ya que un gran nivel de complejidad puede disminuir la interpretabilidad y la explicabilidad.

Para la selección del conjunto de datos, es importante hacer una revisión y comparación de los diferentes conjuntos de datos RGB disponibles para uso público y los empleados en el estado del arte, teniendo en cuenta la necesidad de trabajar con elementos heterogéneos, etiquetados y con una distribución natural (no artificial). Esto debido a que el desbalance de clases puede reducir el desempeño y aumentar el sesgo del modelo, considerando que los datos ya son objeto de sesgo basal (sesgo implícito en todos los procesos de adquisición y procesamiento). En este sentido fue elegido el conjunto de entrenamiento de la base de datos ImageNet (ILSVRC) compuesto por imágenes etiquetadas y balanceadas, con una distribución de categorías que se consideró como la natural y permaneció invariante en los procesos de entrenamiento y prueba de los modelos.

En complemento, es importante tener en cuenta que para el desarrollo de proyectos con datasets de grandes proporciones, se debe realizar una estimación de la capacidad computacional requerida, ya que, contrario a otros modelos, el modelado por Deep Learning requiere mayor cantidad de recursos, por lo que limitaciones en la capacidad computacional pueden impedir que se lleve a cabo el proyecto con las condiciones adecuadas.

El bajo PSNR de los métodos esteganográficos documentados está relacionado con la baja asimilación del payload detectada en este proyecto, lo que explica por qué el uso de payloads custom (entendidos como no usados en el estado del arte) fue requerido, dada la imprecisión entre el payload deseado en la imagen y el que resulta después de embeber esteganografía con los algoritmos de Fridrich [79]. Lo anterior sustenta por qué se obtuvo mejores resultados con otros métodos esteganográficos distintos a los propuestos por el ya mencionado autor.

Las métricas de desempeño escogidas para la evaluación del modelo de Deep Learning fueron F1 score, precisión, tiempo de entrenamiento y tiempo de estimación, teniendo en cuenta que fueron las más usadas en los trabajos analizados en el estado del arte y que permiten obtener un panorama general del estado final del modelo. Además de compararlo en términos de métricas por evaluación, el análisis de complejidad temporal ayuda a vislumbrar que la detección puede ser llevada a cabo con un coste en tiempo relativamente bajo, si se le compara con el tiempo que le tomaría a un humano realizar dicha identificación.

El modelo de Deep Learning mediante redes neuronales convolucionales tiene una precisión del 50% y un 66% en la métrica de desempeño F1 score, comparables con el modelo de Mamada [24]. Sin embargo, este desempeño está lejos de ser comparable con el obtenido en otros escenarios como el reconocimiento de los latidos del corazón (93%-94%), la segmentación de

imágenes de hábitats (94%-95%) y el reconocimiento de expresiones faciales (sonrisas) independientes del sujeto con detección de rostros (97.6%). Esto se debe a los siguientes factores: a) la alta complejidad temporal del modelo en el entrenamiento, b) la baja interpretabilidad y explicabilidad que impiden una profundización en los procesos de mejora, y c) la naturaleza del procedimiento esteganográfico que modifica los pixeles con un patrón distinto al meramente gráfico. En consecuencia, a pesar de que los modelos de Deep Learning son extremadamente útiles en escenarios en donde la detección de objetos gráficos es la prioridad, se trata de modelos poco interpretables que deben usarse bajo circunstancias determinadas, no siendo una de estas la detección de esteganografía LSB.

La realización de experimentos y obtención de resultados adicionales permitió comparar el modelo con otros enfoques en el estado del arte, además de sentar un avance en cuanto a la propuesta de modelos para solución de problemas que involucren contenido multimedia, ya que estos problemas se pueden resolver con modelos más interpretables y menos complejos. Se concluye que la solución del problema propuesto puede ser llevada a cabo por modelos menos complejos, optimizando así el tiempo de desarrollo empleado por el investigador. En complemento, los atributos García mostraron una mejora con respecto a los atributos documentados por Miranda et al., debido a que algunos datos de la media geométrica implementada por Miranda et al. llegan a ser cero, perjudicando así el desempeño del modelo, sentando así un avance importante para el Estado del Arte.

9 RECOMENDACIONES Y TRABAJOS FUTUROS

9.1. Recomendaciones

Usualmente los problemas de Deep Learning pueden ser resueltos por modelos más sencillos, por lo que se recomienda que se evalúe cuidadosamente el costo/beneficio en cada proyecto e iniciar el modelado en orden de complejidad (desde el menos complejo y más interpretable, hasta el más complejo y menos interpretable).

Por otro lado, se deben tener en cuenta las variaciones en la profundidad de color, almacenamiento, estructuración del dataset y método de esteganografía aplicado, ya que la esteganografía no suele ser embebida en forma de bordes ni patrones pictográficos específicos, situación que puede variar con respecto a la implementación del método esteganográfico. Se recomienda estudiar el método esteganográfico, antes de proponer técnicas de detección de caja negra que no puedan ser parametrizadas por su complejidad.

Finalmente, se sugiere estructurar un conjunto de datos por tuplas (cover y stego) para cada imagen de la muestra seleccionada, además almacenar los conjuntos de datos en formato sin compresión, así se podría evitar la alteración de las transformaciones espaciales efectuadas por los métodos esteganográficos.

9.2. Trabajos futuros

Con base en los resultados presentados, se proponen los siguientes trabajos futuros:

- Reestructurar la arquitectura del conjunto de datos usado, para así reforzar el aprendizaje realizado por los distintos modelos sobre el conjunto de datos. A la vez, probar enfoques con diferentes formatos para el almacenamiento de las imágenes.
- Replicar el proceso detallado en este proyecto utilizando algoritmos esteganográficos alternativos embebidos en imágenes para encontrar resultados acordes con el Estado del Arte.
- Proponer una arquitectura de modelo para poder identificar la cantidad de payload y el método esteganográfico presente.
- Desarrollar un modelo de Deep Learning que no sólo se encargue de detectar la existencia de carga esteganográfica, sino que también pueda extraer y decodificar el mensaje oculto que se encuentra embebido.
- Utilizar diferentes enfoques de Deep Learning, como lo son redes neuronales adversariales y redes neuronales profundas, para probar la efectividad de diferentes métodos de Machine Learning en la solución de un mismo problema.
- Extraer los pesos asignados a las neuronas de los modelos de Deep Learning en sus capas convolucionales y densamente conectadas, con el fin de reconstruir los patrones de detección de esteganografía.

10 BIBLIOGRAFÍA

[1] C. Greenwood, “9 subtle ways technology is making humanity worse,” *Business Insider*. https://www.businessinsider.com/technology-negative-bad-effects-society-2019-8 (accessed Aug. 16, 2021).

[2] B. Armstrong, “The Pros And Cons Of Genetically Engineering Humans,” *Medium*, Jun. 09, 2021. https://barmstrong.medium.com/the-pros-and-cons-of-genetically-engineering-humans-49973778c349 (accessed Aug. 16, 2021).

[3] “What are the advantages and disadvantages of artificial intelligence?,” *ResearchGate*. https://www.researchgate.net/post/What-are-the-advantages-and-disadvantages-of-artificial-intelligence (accessed Aug. 16, 2021).

[4] N. Hamid, A. Yahya, R. B. Ahmad, and O. Al-qershi, “Image Steganography Techniques: An Overview,” *Int. J. Comput. Sci. Secur.*, vol. 6, pp. 168–187, Jun. 2012.

[5] McAfee, “McAfee Labs Threats Report.” Jun. 2017. Accessed: Aug. 02, 2020. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2017.pdf

[6] SANS institute, “SANS Institute: Reading Room - Steganography,” *sans.org*. https://www.sans.org/reading-room/whitepapers/stenganography/paper/1014 (accessed Aug. 06, 2020).

[7] K. Karampidis, E. Kavallieratou, and G. Papadourakis, “A review of image steganalysis techniques for digital forensics,” *J. Inf. Secur. Appl.*, vol. 40, pp. 217–235, Jun. 2018, doi: 10.1016/j.jisa.2018.04.005.

[8] Kaspersky, “Steganography: Multiple Hacking Groups are Increasingly Using the Technique to Hide Stolen Information inside Images | Kaspersky,” Aug. 07, 2017. https://www.kaspersky.co.uk/about/press-releases/2017_steganography-multiple-hacking-groups-are-increasingly-using-the-technique-to-hide-stolen-information-inside-images (accessed Aug. 07, 2020).

[9] McAfee, “Protecting Against Steganographic Threats Solution Brief.” Jun. 2017.

[10] “CVE - Search Results.” https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=jpeg (accessed Sep. 23, 2021).

[11] “Zero Day Initiative — CVE-2020-0729: Remote Code Execution Through .LNK Files,” *Zero Day Initiative*. https://www.thezdi.com/blog/2020/3/25/cve-2020-0729-remote-code-execution-through-lnk-files (accessed Sep. 23, 2021).

[12] “CVE-2009-4764 : Adobe Reader 8.x and 9.x on Windows is able to execute EXE files that are embedded in a PDF document, which makes it eas.” https://www.cvedetails.com/cve/CVE-2009-4764/ (accessed Sep. 23, 2021).

[13] J. Jarvis, “Steganography: Combatting Threats Hiding in Plain Sight,” *Fortinet Blog*, Feb. 21, 2018. https://www.fortinet.com/blog/threat-research/steganography--combatting-threats-hiding-in-plain-sight.html (accessed Sep. 23, 2021).

[14] J. Fridrich, M. Goljan, and R. Du, “Reliable Detection of LSB Steganography in Color and Grayscale Images,” *Proc. 2001 Workshop Multimed. Secur. New Chall.*, Oct. 2002, doi: 10.1145/1232454.1232466.

[15] W. Songtao, S. Zhong, and Y. Liu, “Steganalysis via Deep Residual Network,” Dec. 2016, pp. 1233–1236. doi: 10.1109/ICPADS.2016.0167.

[16] S. Wu, S. Zhong, and Y. Liu, “Deep residual learning for image steganalysis,” *Multimed. Tools Appl.*, vol. 77, no. 9, pp. 10437–10453, May 2018, doi: 10.1007/s11042-017-4440-4.
[17] T. I. Team, “New steganography attack targets Azerbaijan,” *Malwarebytes Labs*, Mar. 05, 2021. https://blog.malwarebytes.com/threat-analysis/2021/03/new-steganography-attack-targets-azerbaijan/ (accessed Aug. 15, 2021).
[18] “Steganography in attacks on industrial enterprises (updated) | Kaspersky ICS CERT,” *Kaspersky ICS CERT | Kaspersky Industrial Control Systems Cyber Emergency Response Team*, Jun. 17, 2020. https://ics-cert.kaspersky.com/reports/2020/06/17/steganography-in-attacks-on-industrial-enterprises/ (accessed Sep. 23, 2021).
[19] S. Wu, S. Zhong, Y. Liu, and M. Liu, “CIS-Net: A Novel CNN Model for Spatial Image Steganalysis via Cover Image Suppression,” *ArXiv191206540 Cs Eess*, Dec. 2019, Accessed: Aug. 18, 2020. [Online]. Available: http://arxiv.org/abs/1912.06540
[20] J. Butora and J. Fridrich, “Detection of Diversified Stego Sources with CNNs,” Jan. 2019, vol. 2019, pp. 534–1. doi: 10.2352/ISSN.2470-1173.2019.5.MWSF-534.
[21] Y. Qian, J. Dong, W. Wang, and T. Tan, “Learning and transferring representations for image steganalysis using convolutional neural network,” Sep. 2016, pp. 2752–2756. doi: 10.1109/ICIP.2016.7532860.
[22] J.-F. Couchot, R. Couturier, C. Guyeux, and M. Salomon, “Steganalysis via a Convolutional Neural Network using Large Convolution Filters,” May 2016.
[23] S. Kang, H. Park, and J.-I. Park, “CNN-Based Ternary Classification for Image Steganalysis,” *Electronics*, vol. 8, p. 1225, Oct. 2019, doi: 10.3390/electronics8111225.
[24] N. Mamada, “Image Steganalysis with Very Deep Convolutional Neural Networks,” 2019.
[25] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, p. 60, Jul. 2019, doi: 10.1186/s40537-019-0197-0.
[26] J. Kim, H. Park, and J.-I. Park, “CNN-based image steganalysis using additional data embedding,” *Multimed. Tools Appl.*, vol. 79, no. 1, pp. 1355–1372, Jan. 2020, doi: 10.1007/s11042-019-08251-3.
[27] Z. Wang, M. Chen, Y. Yang, M. Lei, and Z. Dong, “Joint multi-domain feature learning for image steganalysis based on CNN,” *EURASIP J. Image Video Process.*, vol. 2020, no. 1, p. 28, Jul. 2020, doi: 10.1186/s13640-020-00513-7.
[28] Z. Jin, Y. Yang, Y. Chen, and Y. Chen, “IAS-CNN: Image adaptive steganalysis via convolutional neural network combined with selection channel,” *Int. J. Distrib. Sens. Netw.*, vol. 16, no. 3, p. 1550147720911002, Mar. 2020, doi: 10.1177/1550147720911002.
[29] J. Miranda and D. Parada, *LSB Steganography Detection in Monochromatic Still Images using Artificial Neural Networks*. 2020.
[30] J. Miranda, “Steganalysis for still images with LSB Steganography - Features dataset.” IEEE, Nov. 21, 2019. Accessed: Jan. 31, 2021. [Online]. Available: https://ieee-dataport.org/open-access/steganalysis-still-images-lsb-steganography-features-dataset
[31] S. Kang, H. Park, and J.-I. Park, “Identification of Multiple Image Steganographic Methods Using Hierarchical ResNets,” *IEICE Trans. Inf. Syst.*, vol. E104.D, no. 2, pp. 350–353, 2021, doi: 10.1587/transinf.2020EDL8116.
[32] G. Xu, H.-Z. Wu, and Y.-Q. Shi, “Structural Design of Convolutional Neural Networks for Steganalysis,” *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 708–712, May 2016, doi: 10.1109/LSP.2016.2548421.
[33] “Definition of STEGANOGRAPHY.” https://www.merriam-webster.com/dictionary/steganography (accessed Aug. 15, 2021).

[34] J. Fridrich, M. Goljan, and D. Soukal, “Searching for the stego-key,” Jun. 2004, vol. 5306, pp. 70–82. doi: 10.1117/12.521353.
[35] “LokiBot Malware | CISA.” https://us-cert.cisa.gov/ncas/alerts/aa20-266a (accessed Aug. 15, 2021).
[36] “Steganography Anchors Pinpoint Attacks on Industrial Targets.” https://threatpost.com/steganography-pinpoint-attacks-industrial-targets/156151/ (accessed Aug. 15, 2021).
[37] book-info com – S. Hendel, *Color in Business, Science, and Industry [Wiley-Interscience: Third edition]*. Accessed: Aug. 21, 2021. [Online]. Available: https://www.book-info.com/isbn/0-471-45212-2.htm
[38] J. Abraham and V. Paul, “An Imperceptible Spatial Domain Color Image Watermarking Scheme,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 31, Dec. 2016, doi: 10.1016/j.jksuci.2016.12.004.
[39] A. Almaliki, R. Din, and M. Mahmuddin, “Review on Steganography Methods in Multi-Media Domain,” pp. 288–292, Aug. 2019.
[40] N. Akhtar, V. Ahamad, and H. Javed, “A compressed LSB steganography method,” in *2017 3rd International Conference on Computational Intelligence Communication Technology (CICT)*, Feb. 2017, pp. 1–7. doi: 10.1109/CIACT.2017.7977371.
[41] J. Deng, Y. Zhong, Z. Wei, and G. Wei, “A Study of Information Hiding Algorithm Based on the Human Vision,” 2015. doi: 10.2991/ITMS-15.2015.343.
[42] C.-H. J. Codr, “Unseen: An Overview of Steganography and Presentation of Associated Java Application C-Hide,” 2009. https://www.semanticscholar.org/paper/Unseen%3A-An-Overview-of-Steganography-and-of-Java-Codr/97c3fe437064594be78605f15218876aabc53e62 (accessed Aug. 16, 2021).
[43] “Re: Human eye color change sensibility in CIELAB units.,” *researchgate*, Jun. 17, 2014. https://www.researchgate.net/post/Human_eye_color_change_sensibility_in_CIELAB_units/53a00fa8d11b8bfe6a8b45e8/citation/download (accessed Sep. 07, 2021).
[44] X.-D. Zhang, “Machine Learning,” in *A Matrix Algebra Approach to Artificial Intelligence*, X.-D. Zhang, Ed. Singapore: Springer, 2020, pp. 223–440. doi: 10.1007/978-981-15-2770-8_6.
[45] “Supervised Learning - an overview | ScienceDirect Topics.” https://www.sciencedirect.com/topics/computer-science/supervised-learning (accessed Sep. 05, 2021).
[46] W. Zhang and H.-M. Sun, “Instagram Spam Detection,” in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, Jan. 2017, pp. 227–228. doi: 10.1109/PRDC.2017.43.
[47] Z. Ghahramani, “Unsupervised Learning,” in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds. Berlin, Heidelberg: Springer, 2004, pp. 72–112. doi: 10.1007/978-3-540-28650-9_5.
[48] “Unsupervised Clustering Approach for Network Anomaly Detection | SpringerLink.” https://link.springer.com/chapter/10.1007/978-3-642-30507-8_13 (accessed Sep. 05, 2021).
[49] X. Zhu and A. B. Goldberg, “Introduction to Semi-Supervised Learning,” *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 3, no. 1, pp. 1–130, Jan. 2009, doi: 10.2200/S00196ED1V01Y200906AIM006.

[50] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, “Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, Dec. 2020, doi: 10.1109/TVT.2020.3034800.
[51] C. Amato and G. Shani, “High-level reinforcement learning in strategy games,” Jan. 2010, vol. 1, pp. 75–82. doi: 10.1145/1838206.1838217.
[52] I. H. Sarker, “Machine Learning: Algorithms, Real-World Applications and Research Directions,” *SN Comput. Sci.*, vol. 2, no. 3, p. 160, Mar. 2021, doi: 10.1007/s42979-021-00592-x.
[53] “CS109.” https://web.stanford.edu/class/archive/cs/cs109/cs109.1178/ (accessed Sep. 07, 2021).
[54] L. Su, “Logistic Regression, Accuracy, Cross-Validation,” *Medium*, May 14, 2019. https://medium.com/@lily_su/logistic-regression-accuracy-cross-validation-58d9eb58d6e6 (accessed Sep. 05, 2021).
[55] A. F. Schmidt and C. Finan, “Linear regression and the normality assumption,” *J. Clin. Epidemiol.*, vol. 98, pp. 146–151, Jun. 2018, doi: 10.1016/j.jclinepi.2017.12.006.
[56] “Sigmoid Function,” *DeepAI*, Sep. 27, 2020. https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function (accessed Sep. 08, 2021).
[57] B. Zoph and Q. V. Le, “Neural Architecture Search with Reinforcement Learning,” *ArXiv161101578 Cs*, Feb. 2017, Accessed: Jan. 20, 2021. [Online]. Available: http://arxiv.org/abs/1611.01578
[58] F. Bre, J. Gimenez, and V. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks,” *Energy Build.*, vol. 158, Nov. 2017, doi: 10.1016/j.enbuild.2017.11.045.
[59] A. Acharya, A. Prakash, P. Saxena, and A. Nigam, “Sampling: Why and How of it? Anita S Acharya, Anupam Prakash, Pikee Saxena, Aruna Nigam,” *Indian J. Med. Specilaities*, Jan. 2013, doi: 10.7713/ijms.2013.0032.
[60] “Stratified sampling in Machine Learning. | by Saaransh Menon | Analytics Vidhya | Medium.” https://medium.com/analytics-vidhya/stratified-sampling-in-machine-learning-f5112b5b9cfe (accessed Sep. 07, 2021).
[61] M. Grootendorst, “Validating your Machine Learning Model,” *Medium*, May 15, 2020. https://towardsdatascience.com/validating-your-machine-learning-model-25b4c8643fb7 (accessed Sep. 07, 2021).
[62] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.
[63] “Beyond Manual Tuning of Hyperparameters | SpringerLink.” https://link.springer.com/article/10.1007/s13218-015-0381-0 (accessed Sep. 07, 2021).
[64] P. Liashchynskyi and P. Liashchynskyi, “Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS,” *ArXiv191206059 Cs Stat*, Dec. 2019, Accessed: Sep. 07, 2021. [Online]. Available: http://arxiv.org/abs/1912.06059
[65] S. Falkner, A. Klein, and F. Hutter, “BOHB: Robust and Efficient Hyperparameter Optimization at Scale,” *ArXiv180701774 Cs Stat*, Jul. 2018, Accessed: Sep. 07, 2021. [Online]. Available: http://arxiv.org/abs/1807.01774
[66] “An overview of gradient descent optimization algorithms.” https://ruder.io/optimizing-gradient-descent/ (accessed Sep. 07, 2021).

[67] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,” *ArXiv14047828 Cs*, Oct. 2014, doi: 10.1016/j.neunet.2014.09.003.
[68] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, May 2019, doi: 10.1038/s42256-019-0048-x.
[69] A. Kamilaris and F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” *Comput. Electron. Agric.*, vol. 147, pp. 70–90, Apr. 2018, doi: 10.1016/j.compag.2018.02.016.
[70] S. Christin, É. Hervet, and N. Lecomte, “Applications for deep learning in ecology,” *Methods Ecol. Evol.*, vol. 10, no. 10, pp. 1632–1644, 2019, doi: 10.1111/2041-210X.13256.
[71] C. Cao *et al.*, “Deep Learning and Its Applications in Biomedicine,” *Genomics Proteomics Bioinformatics*, vol. 16, no. 1, pp. 17–32, Feb. 2018, doi: 10.1016/j.gpb.2017.07.003.
[72] “Deep Learning in Chemistry | Journal of Chemical Information and Modeling.” https://pubs.acs.org/doi/abs/10.1021/acs.jcim.9b00266 (accessed Sep. 07, 2021).
[73] “Redes neuronales convolucionales,” *IBM Developer*, Dec. 07, 2018. https://developer.ibm.com/es/technologies/artificial-intelligence/articles/cc-convolutional-neural-network-vision-recognition/ (accessed Jan. 20, 2021).
[74] B. AI, “Redes neuronales convolucionales,” *Medium*, Nov. 27, 2019. https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8 (accessed Jan. 20, 2021).
[75] “Convolution - an overview | ScienceDirect Topics.” https://www.sciencedirect.com/topics/mathematics/convolution (accessed Aug. 16, 2021).
[76] C. Baskin, N. Liss, A. Mendelson, and E. Zheltonozhskii, “Streaming Architecture for Large-Scale Quantized Neural Networks on an FPGA-Based Dataflow Platform,” Jul. 2017.
[77] A. Kakde, D. Sharma, B. Kaushik, and N. Arora, “Investigation of Solar Flare Classification to Identify Optimal Performance,” *Electron. Lett. Comput. Vis. Image Anal.*, vol. 20, pp. 28–41, Jan. 2021, doi: 10.5565/rev/elcvia.1274.
[78] Ó. P. Montoya and L. B. Molina, “REDES NEURONALES CONVOLUCIONALES PROFUNDAS PARA EL RECONOCIMIENTO DE EMOCIONES EN IMÁGENES,” p. 64.
[79] “Understanding of a convolutional neural network - IEEE Conference Publication.” https://ieeexplore.ieee.org/abstract/document/8308186 (accessed Jan. 21, 2021).
[80] L. M. Surhone, M. T. Tennoe, and S. F. Henssonow, *Softmax Activation Function*. Betascript Publishing, 2010.
[81] “Convolutional neural network,” *Wikipedia*. Aug. 12, 2021. Accessed: Aug. 16, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=1038482044
[82] “Why is Python the Best-Suited Programming Language for Machine Learning?,” *GeeksforGeeks*, Aug. 27, 2019. https://www.geeksforgeeks.org/why-is-python-the-best-suited-programming-language-for-machine-learning/ (accessed Sep. 08, 2021).
[83] S. Baluja, “Hiding Images in Plain Sight: Deep Steganography,” p. 11.
[84] “Convertir imagen RGB o colores a escala de grises - MATLAB rgb2gray - MathWorks América Latina.” https://la.mathworks.com/help/matlab/ref/rgb2gray.html (accessed Sep. 06, 2021).
[85] “Steganographic Algorithms.” http://dde.binghamton.edu/download/stego_algorithms/ (accessed Aug. 15, 2021).

[86] A. Almohammad and G. Ghinea, “Stego image quality and the reliability of PSNR,” in *2010 2nd International Conference on Image Processing Theory, Tools and Applications*, Jul. 2010, pp. 215–220. doi: 10.1109/IPTA.2010.5586786.

[87] S. Gumhold, X. Wang, and R. MacLeod, “Feature Extraction from Point Clouds,” *Proc. 10th Int. Meshing Roundtable*, vol. 2001, Nov. 2001.

[88] “Feature extraction from faces using deformable templates | SpringerLink.” https://link.springer.com/article/10.1007%2FBF00127169 (accessed Sep. 07, 2021).

[89] Z.-Q. Hong, “Algebraic feature extraction of image for recognition,” *Pattern Recognit.*, vol. 24, no. 3, pp. 211–219, Jan. 1991, doi: 10.1016/0031-3203(91)90063-B.

[90] D. D. Lewis, “Feature selection and feature extraction for text categorization,” in *Proceedings of the workshop on Speech and Natural Language - HLT ’91*, Harriman, New York, 1992, p. 212. doi: 10.3115/1075527.1075574.

[91] T. Musha, Y. Terasaki, H. A. Haque, and G. A. Ivamitsky, “Feature extraction from EEGs associated with emotions,” *Artif. Life Robot.*, vol. 1, no. 1, pp. 15–19, Mar. 1997, doi: 10.1007/BF02471106.

11 ANEXOS

Anexo 1. Evidencias ciclo de trabajo SCRUM.
Se documentaron 17 sprints de desarrollo, los cuales son descritos a continuación.

Sprint	Prototipos/experimentos	Backlog
1		Discusión sobre aspectos iniciales del desarrollo - División del dataset y aumento del almacenamiento
2		Discusión sobre el preprocesamiento del dataset
3		Discusión sobre la adquisición y pre-procesamiento del dataset
4		Discusión sobre la adquisición y pre-procesamiento del dataset
5		Discusión sobre esteganografía y algoritmos de detección
6	PF1, PF2	Discusión sobre CNN
7	PF3	Disusión sobre esteganografía y Big Data - Dask, PySpark, Numba
8	PF4	Discusión resultados de esteganografía y detección con CNN imágenes monocromáticas - Big Data
9	PF5	Discusión resultados de procesamiento de Big Data
10		Discusión sobre EDA de los algoritmos utilizados
11	P6	Discusión sobre iteración en la selección de algoritmos a utilizar
12	P5, P1	Discusión sobre prototipos obtenidos y documentación
13	P2 – P4	Discusión sobre prototipos obtenidos y propuesta de nuevos enfoques como experimentos
14	E6, E7	Discusión sobre avances en documentación y experimentos
15	E1-E5	Discusión sobre contraste de prototipos con otros modelos y resultados
16		Discusión sobre resultados obtenidos y Estado del Arte
17		Discusión sobre documentación final

Anexo 2. Acceso al repositorio.

Para acceder se debe ingresar al siguiente enlace con un correo dentro del dominio upb.edu.co: https://upbeduco-my.sharepoint.com/:f:/g/personal/alberto_garcia_2016_upb_edu_co/En2IKpmlpX9CoMUyaYALgTkBZQvL_KDVynBsJDhEu7YehQ?e=8dnsfg

Anexo 3. Prototipos fallidos.

En este anexo se mencionará algunos prototipos que, por capacidad computacional o técnica, no funcionaron correctamente.

1. Prototipo de CNN con imágenes a color de 256*256, en este se estableció la estructura principal de la CNN, no se puedo leer el dataset, debido a la capacidad computacional disponible.

Capas	Neuronas por capa	Tamaño del kernel	Activación
Input	-	-	-
Conv2D	256 y 512	3	ReLU
MaxPooling2D	-	3	-
Flatten	-	-	-
Dense	512 y 256	-	ReLU
Output	1	-	Softmax

2. Prototipo de CNN con imágenes a color de 128*128. No se pudo ejecutar la lectura de datos en el VPS suministrado, debido a la poca memoria disponible.

Capas	Neuronas por capa	Tamaño del kernel	Activación
Input	-	-	-
Conv2D	128, 256 y 512	3	ReLU
MaxPooling2D	-	3	-
Flatten	-	-	-
Dense	512, 256 y 128	-	ReLU
Output	1	-	Softmax

3. Prototipo de CNN con imágenes a color de 128*128 y cambio en la estructura para el almacenamiento del dataset, no fue posible leer los datos. Se agregaron más capas al prototipo. Comparte arquitectura con el siguiente prototipo.

4. Prototipo de CNN con imágenes a color de 128*128 y herramientas de Big Data para lectura del dataset. No se pudo realizar el entrenamiento debido a cierta incompatibilidad

entre la librería para trabajos de Big Data y el modelo CNN. Se agregó proceso de K-Folds.

Capas	Neuronas por capa	Tamaño del kernel	Activación
Input	-	-	-
Conv2D	128, 256 y 512	3	ReLU
MaxPooling2D	-	3	-
Flatten	-	-	-
Dense	512, 256 y 128	-	ReLU
Output	1	-	Softmax

5. Prototipo de CNN con un solo canal de las imágenes 128*128. No se pudo realizar el entrenamiento debido a limitaciones en la capacidad de cómputo.

Capas	Neuronas por capa	Tamaño del kernel	Activación
Input	-	-	-
Conv2D	128, 256, 512 y 1024	5	ReLU
MaxPooling2D	-	3	-
Flatten	-	-	-
Dense	1024, 512, 256, 128, 8 y 4	-	ReLU
Output	2	-	Softmax