Soria, Karen Micaela

Detección de patrones de Phishing en imágenes mediante técnicas de Deep Learning

2024

Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

INSTITUTO DE INGENIERÍA Y AGRONOMÍA

INGENIERÍA EN INFORMÁTICA

PRÁCTICA PROFESIONAL SUPERVISADA

Informe Final

Detección de patrones de Phishing en imágenes mediante técnicas de Deep Learning

KAREN MICAELA SORIA

FLORENCIO VARELA, 18 DE DICIEMBRE DE 2024

Resumen

Este trabajo presenta una solución basada en técnicas de Deep Learning para detectar patrones visuales en imágenes asociadas a phishing, una de las principales amenazas en ciberseguridad. Utilizando la arquitectura VGG16 basada en redes neuronales convolucionales (CNN), se implementó y evaluó un modelo capaz de identificar sitios maliciosos, alcanzando una precisión cercana al 90%.

El estudio destaca la importancia de contar con conjuntos de datos balanceados para evitar sesgos y mejorar la capacidad de generalización del modelo. Durante la investigación se abordaron desafíos como la detección de imágenes ambiguas y las limitaciones computacionales para realizar el entrenamiento.

El modelo desarrollado representa un aporte en la lucha contra el phishing y sienta las bases para futuras mejoras, como la ampliación del dataset, la optimización del modelo y su implementación en sistemas de tiempo real.

Palabras Clave: Aprendizaje profundo, Phishing, Redes Neuronales Convolucionales, VGG16, Ciberseguridad, Detección de patrones visuales.

Abstract

This work presents a Deep Learning-based solution for detecting visual patterns in images associated with phishing, one of the most prevalent threats in cybersecurity. Using the VGG16 convolutional neural network (CNN) architecture, a model was implemented and trained to identify malicious sites by analyzing their visual characteristics. The results achieved demonstrate a precision of nearly 90%, validating the effectiveness of the proposed approach.

The study highlights the importance of using balanced datasets to improve model generalization and reduce prediction bias. While the model performed well on complex phishing images, challenges were identified, particularly in handling ambiguous images visually similar to legitimate websites and in computational resource limitations.

In conclusion, the developed model represents an advancement in phishing detection. Future work includes expanding the dataset, optimizing the model with advanced architectures, and implementing real-time detection systems to address evolving cyber threats.

Keywords: Deep Learning, Phishing, Convolutional Neural Networks, VGG16, Cybersecurity, Visual Pattern Detection.

Dedicatoria y agradecimientos

Quiero expresar mi profundo agradecimiento a la Universidad Nacional Arturo Jauretche por acercarme la posibilidad de estudiar esta hermosa carrera. A los docentes, cuyo entusiasmo por la enseñanza y capacidad para transmitir conocimientos de manera única han sido una fuente constante de inspiración.

En especial, agradezco a mi tutor Ing. Carlos Schenone por su increíble acompañamiento y buena predisposición en cada paso de este proyecto. Además al coordinador de la carrera Dr. Ing. Martin Morales por brindarme esta oportunidad y por su compromiso continuo con la mejora de la calidad educativa.

Dedico este logro a mi familia, cuyo apoyo incondicional ha sido fundamental para llegar hasta acá. A mis compañeros y futuros colegas, quienes me acompañaron a lo largo de este camino, y a mis amigos, que fueron un gran alivio en los momentos más desafiantes. A todos ustedes, gracias por ser parte de este importante capítulo de mi vida.

Índice de contenido

Índice de Figuras

1. Introducción

El presente trabajo forma parte de la Práctica Profesional Supervisada (PPS) de la carrera Ingeniería en Informática de la Universidad Nacional Arturo Jauretche (UNAJ), siendo parte de los requisitos para la obtención del título.

En un contexto donde las amenazas cibernéticas son cada vez más sofisticadas el objetivo general de esta investigación es desarrollar una solución basada en técnicas de Deep Learning que permita la detección de patrones visuales compatibles con ataques de phishing a partir del análisis de imágenes, contribuyendo a fortalecer las medidas de ciberseguridad en las etapas tempranas de detección de amenazas y prevención de incidentes.

Las técnicas de inteligencia artificial han demostrado logros en diversos campos, entre ellos la ciberseguridad, donde ciertas actividades maliciosas como la suplantación de identidad, entre las cuales destaca el phishing, pueden ser identificadas a partir del análisis de las imágenes de los sistemas vulnerados [1].

Considerando lo anterior, las actividades desarrolladas en la práctica se enfocarán en investigar y desarrollar un clasificador basado en redes neuronales entrenado para identificar patrones compatibles con phishing en imágenes mediante algoritmos de Deep Learning. La investigación a desarrollar se enfocará en el diseño y construcción de un modelo basado en redes neuronales profundas alimentado con un conjunto de datos conformado por imágenes de sitios web correspondientes a phishing y sitios normales. Durante el entrenamiento y validación se utilizarán técnicas específicas para el procesamiento de imágenes, luego se evaluará su desempeño utilizando métricas específicas como la precisión y sensibilidad.

El tema de estudio seleccionado tiene gran relevancia en el contexto actual de la seguridad informática, donde la prevención de ataques cibernéticos es una necesidad primordial con el objetivo de detener o al menos frenar su incremento sostenido [2]. En este grupo, destacan los ataques compatibles con técnicas de phishing, representando el sesenta por ciento de los incidentes de seguridad reportados globalmente según un estudio de la empresa Verizon [3]. Siguiendo la tendencia, en Argentina los ataques del

tipo phishing se ubican en primer lugar, como se puede observar en la imagen siguiente obtenida el reporte anual para el año 2023 del equipo local de respuesta ante incidentes (ARCERT).

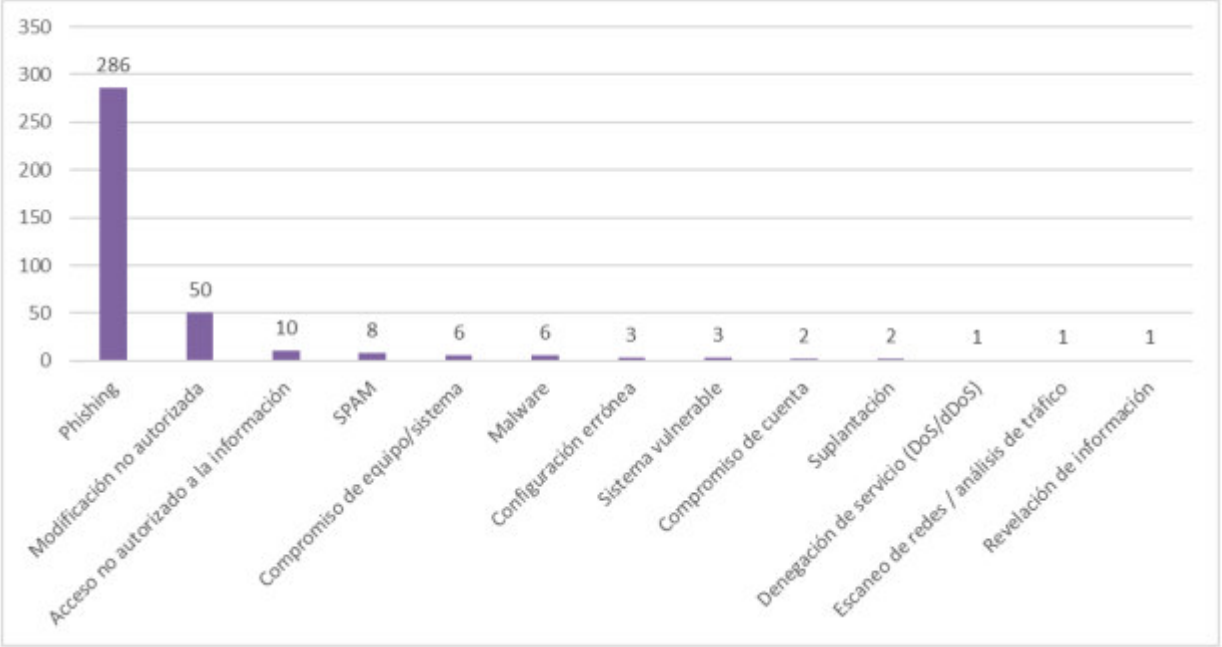

Figura 1. Informe Anual de Incidentes de Seguridad registrados en el 2023 por el equipo de respuesta ante Emergencias Informáticas Nacional, [en línea] Disponible*:* *https://www.argentina.gob.ar/sites/default/files/2024/05/informe_2023_del_cert.ar_.pdf* (Accedido el 13 de noviembre de 2024)

En este contexto, una herramienta automatizada para la detección y alerta de phishing fortalece los sistemas colaborando sobre todo con aquellos sectores donde la protección de datos es crítica para mantener la confianza de los usuarios, a la vez que realiza un aporte al desarrollo de sistemas que contribuyan a la promoción de un entorno digital más seguro.

1.1 Motivación

La motivación para esta investigación radica en la necesidad de fortalecer la detección de amenazas cibernéticas mediante el análisis de imágenes, con un enfoque particular en el uso de técnicas avanzadas de aprendizaje profundo. Este trabajo se centra en el desarrollo de métodos que identifican patrones visuales asociados a actividades maliciosas, específicamente ataques de phishing, el cual se presenta como uno de los métodos más utilizados y efectivos para acceder a información sensible de los usuarios, logrando engañar a las víctimas al hacerse pasar por entidades confiables a través de sitios web falsos [4].

A pesar de los esfuerzos por mejorar los mecanismos de defensa mediante el fortalecimiento de las políticas de seguridad y la sensibilización de los usuarios, el phishing continúa adaptándose a las nuevas tecnologías y superando las técnicas de detección actuales. Frente a esta problemática, se plantea la necesidad de explorar nuevas soluciones tecnológicas que puedan ofrecer una respuesta efectiva ante estas amenazas.

En este contexto, surge la pregunta de investigación: **¿Puede el Deep Learning fortalecer la seguridad informática y contribuir activamente a la prevención del phishing mediante herramientas innovadoras?**

Este proyecto busca responder a esta cuestión mediante la aplicación de algoritmos de aprendizaje profundo con el propósito de identificar patrones compatibles con un ataque de phishing en imágenes.

El aprendizaje profundo, como subcampo de la inteligencia artificial, ha mostrado un gran potencial en el análisis de grandes volúmenes de datos, permitiendo detectar características visuales que, en muchos casos, pueden pasar desapercibidas para el ojo humano [5]. En este sentido, la capacidad del Deep Learning para reconocer patrones complejos representa una oportunidad para mejorar la detección y prevención de ataques de phishing, facilitando el desarrollo de herramientas automatizadas que brinden una mayor seguridad informática [6, 7].

1.2 Objetivos

El objetivo principal de este trabajo es desarrollar un sistema de detección de patrones compatibles con phishing en imágenes mediante redes neuronales convolucionales (CNN). El sistema entregará las imágenes provenientes de capturas de pantalla a un modelo basado en redes neuronales convolucionales, cuyo objetivo es clasificar la entrada como “phishing” o “normal”. El trabajo busca contribuir no solo con una solución tecnológica al problema del phishing, sino también con el fortalecimiento de la ciberseguridad en un entorno digitalizado.

1.2.1 Objetivos Específicos

Para lograr el objetivo principal se han planteado los siguientes objetivos específicos, destinados a orientar las actividades de investigación y desarrollo del sistema:

Diseñar y optimizar un modelo de Deep Learning para la detección de actividades maliciosas compatibles con phishing

- Diseñar y entrenar un modelo de CNN con capacidad para extraer características visuales distintivas que sean relevantes para la detección de ataques de phishing.
- Evaluar diferentes arquitecturas de redes neuronales y técnicas de optimización para identificar la configuración que ofrezca el mejor rendimiento.

Desarrollar un sistema de clasificación de patrones visuales en imágenes.

- Identificar y clasificar patrones visuales específicos asociados con actividades maliciosas compatibles con phishing, tales como anomalías en las interfaces de usuario.
- Implementar algoritmos de clasificación para mejorar la precisión en la detección y diferenciación de estos patrones.

Preprocesar un conjunto de datos de imágenes etiquetadas para el entrenamiento y evaluación del modelo.

- Recolectar un conjunto de imágenes relevantes y etiquetarlas con información sobre las actividades maliciosas detectadas, compatibles con los ataques de phishing.
- Realizar preprocesamiento de imágenes y dividir el conjunto de datos en conjuntos de entrenamiento y evaluación para facilitar el aprendizaje y análisis del modelo.

Evaluar la eficiencia y desempeño del modelo en la detección de ataques.

- Medir el rendimiento del modelo utilizando métricas estándar, como precisión y sensibilidad, para determinar su efectividad en la clasificación.
- Evaluar los resultados del modelo para identificar mejoras y aplicar los ajustes posibles y realizar propuestas para la continuación del trabajo.

2. Marco Teórico

2.1. Inteligencia Artificial (IA)

Es una rama de la informática que se centra en la creación de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, tales como la interpretación de datos, la toma de decisiones y la resolución de problemas. En términos generales, la IA busca dotar a las máquinas de la capacidad para aprender de su entorno, adaptarse a situaciones nuevas y actuar de manera autónoma, facilitando su aplicación en áreas tan diversas como la medicina, el transporte y la seguridad informática [8].

Los sistemas de IA se dividen en dos categorías principales: IA débil y IA fuerte . La IA débil se limita a realizar tareas específicas, como el reconocimiento de voz o el análisis de patrones en imágenes, en los que un conjunto de algoritmos toma decisiones basadas en reglas predefinidas o datos históricos. Por otro lado, la IA fuerte busca replicar en su totalidad las capacidades cognitivas humanas, a superar habilidades de razonamiento, planificación y comprensión contextual, aunque aún queda un largo camino por recorrer para alcanzar este nivel.

Desde su surgimiento, la IA ha evolucionado significativamente, pasando de sistemas que se basaban en reglas lógicas a métodos que involucran el aprendizaje automático (Machine Learning, ML) y el aprendizaje profundo (Deep Learning, DL) [9]. Esta transición ha marcado un cambio de paradigma en el que los sistemas ahora pueden aprender y mejorar su rendimiento a partir de datos, en lugar de depender únicamente de la programación específica de cada tarea.

Dentro de la IA, el aprendizaje profundo ha demostrado ser especialmente eficaz en el análisis de grandes volúmenes de datos y en el reconocimiento de patrones complejos [10] como es el caso de la detección de amenazas en el ámbito de la ciberseguridad. Por ejemplo, los sistemas de IA débil son ampliamente utilizados en herramientas de monitoreo de tráfico de red y detección de intrusos, mientras que el desarrollo de IA fuerte aún está en fases experimentales, pero promete revolucionar la forma en que se gestionan las amenazas emergentes. Al utilizar redes neuronales artificiales, especialmente las redes neuronales convolucionales (CNN) , el aprendizaje profundo

permite identificar patrones en imágenes que pueden ser indicativos de actividades maliciosas, como los intentos de phishing en capturas de pantalla de sitios web [11].

El avance en IA ha dado lugar a diversos subcampos que enriquecen su aplicación práctica:

- Procesamiento de lenguaje natural (PNL): enfocado en comprender y generar lenguaje humano, facilitando la interacción entre humanos y máquinas y mejorando la capacidad de los sistemas para reconocer intentos de engaño en el texto.
- Visión por computadora: un área clave para el análisis de imágenes, donde las CNN juegan un papel importante en la identificación de patrones visuales. Esta capacidad resulta esencial en investigaciones donde se requiere que el sistema detecte formas y objetos, por ejemplo logística y conducción autónoma.

- Sistemas expertos: estos sistemas ayudan a reducir la incertidumbre en la toma de decisiones, en áreas como el medioambiente, ofreciendo recomendaciones y alertas basadas en las características extraídas de grandes conjuntos de datos.

Con el avance de la IA, el aprendizaje profundo ha ganado terreno como una herramienta de valor en la seguridad informática, sobre todo en el desarrollo de sistemas que actúen de manera preventiva, colaborando en la detección y alerta temprana sobre posibles amenazas [18], posicionándose como una gran aliada en la lucha contra el fraude digital, aportando soluciones innovadoras para la protección de los usuarios en un entorno digital en constante evolución.

2.2. Aprendizaje automático (ML)

Dentro del ámbito de la Inteligencia Artificial, el Machine Learning (ML) o aprendizaje automático destaca como una subdisciplina clave, centrada en el desarrollo de sistemas que pueden aprender automáticamente a partir de datos sin necesidad de una programación específica para cada tarea [19]. En lugar de seguir instrucciones codificadas paso a paso, los sistemas de ML analizan grandes volúmenes de datos,

identifican patrones y extraen tendencias que luego les permiten realizar predicciones o tomar decisiones de manera autónoma .

El aprendizaje automático supone un cambio en el paradigma de programación tradicional, en lugar de proporcionar reglas y algoritmos específicos, se suministran datos de entrada junto con las respuestas esperadas y el sistema "aprende" a inferir el modelo o las reglas necesarias para obtener las respuestas deseadas. Este enfoque hace posible que los sistemas desarrollen su propia comprensión de las relaciones entre los datos, aplicando ese conocimiento a nuevos datos mejorando su desempeño con la práctica, apoyado por técnicas de análisis estadístico y procesamiento de datos a gran escala, aprovechan la capacidad de cómputo moderno para trabajar con volúmenes de datos masivos que serían inabordables con los métodos estadísticos tradicionales.

Existen tres tipos principales de algoritmos de aprendizaje automático, cada uno definido por su enfoque para entrenar los modelos de acuerdo a los datos disponibles y el tipo de problema a resolver:

Aprendizaje Supervisado: Este tipo de algoritmo se basa en entrenar un modelo con datos de entrada etiquetados, donde cada ejemplo está acompañado de una clasificación correcta o etiqueta. El sistema utiliza estos pares de datos para aprender a mapear entradas con salidas y posteriormente, realizar predicciones sobre datos nuevos no etiquetados. Los campos aplicación incluyen sistemas de clasificación de imágenes y predicción de tendencias.

Aprendizaje No Supervisado: A diferencia del aprendizaje supervisado, en este caso el modelo trabaja con datos no etiquetados y busca patrones característicos en ellos sin contar con una etiqueta o respuesta correcta explícita. El objetivo del algoritmo es descubrir estructuras subyacentes, que le permitan realizar categorizaciones o agrupaciones de los datos. Este abordaje es útil en análisis exploratorios y aplicaciones de marketing como la segmentación de clientes.

Aprendizaje por Refuerzo: El algoritmo se basa en la interacción con un entorno dinámico, donde el sistema aprende a tomar decisiones con el objetivo de maximizar una recompensa acumulada. En esta técnica, el modelo no recibe etiquetas o respuestas

correctas específicas, sino que aprende iterando en un proceso de prueba y error, ajustando sus acciones de acuerdo a las recompensas o penalizaciones recibidas. Se utiliza en aplicaciones que requieren la toma de decisiones en tiempo real, como los sistemas de control para robots.

El aprendizaje automático representa un avance fundamental en la capacidad de los sistemas para analizar y responder a datos complejos. Al aprender directamente de los datos y refinar su comportamiento a medida que recibe información adicional, los modelos se vuelven cada vez más efectivos y precisos, abriendo su aplicación a un grupo amplio de problemas complejos, desde el medio ambiente hasta la medicina [20], aportando soluciones a desafíos que antes resultaban muy difíciles de abordar con la programación tradicional.

2.3. Aprendizaje profundo (DL)

El aprendizaje profundo es una categoría dentro del aprendizaje automático, que se caracteriza por utilizar una arquitectura basada en redes neuronales cuyo modelo se construye a partir de múltiples capas de neuronas interconectadas aplicando un enfoque de procesamiento jerárquico y no lineal, brindando múltiples capas de procesamiento permitiendo que el sistema aprenda representaciones complejas y patrones ocultos en grandes volúmenes de datos. A diferencia de los modelos de aprendizaje automático tradicionales, que suelen requerir una extracción manual de características, en DL el modelo aprende directamente de los datos, sin intervención manual, construyendo una jerarquía de características desde los aspectos básicos hasta los más abstractos.

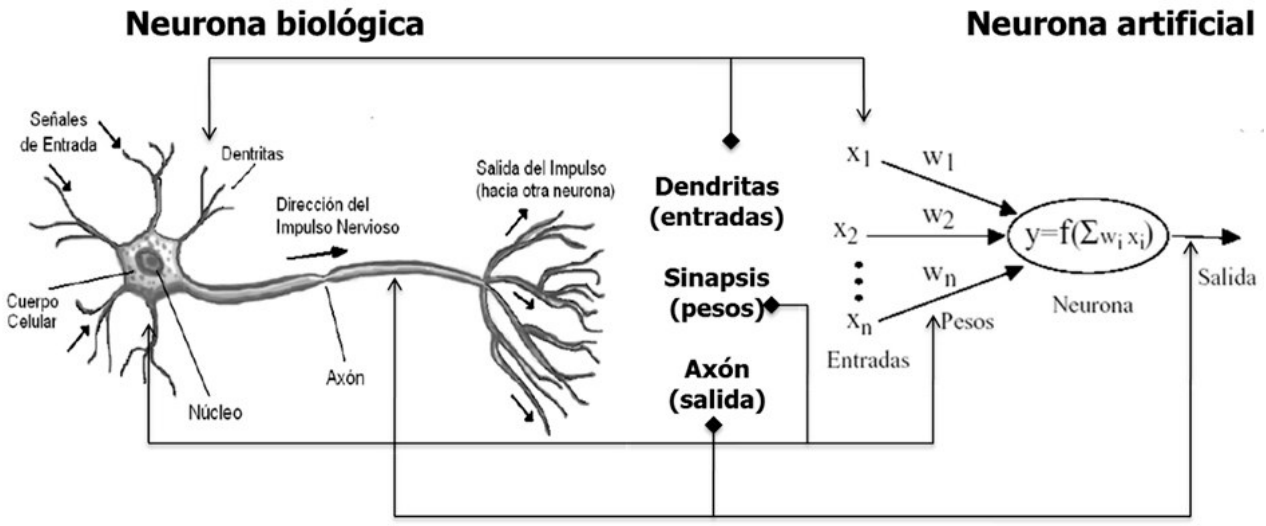

Figura 2. Diagrama de Neurona biológica y neurona artificial, [en línea] Disponible*:* https://www.redalyc.org/journal/1815/181549596004/html/

El modelo de DL está inspirado en el funcionamiento del cerebro humano, imitando cómo las neuronas se conectan entre sí y transmiten información [21]. Cada capa en una red neuronal profunda transforma los datos de entrada, extrayendo patrones más atractivos y específicos a medida que avanzan por las capas. Entre los tipos más comunes de redes neuronales en aprendizaje profundo se destacan las redes neuronales convolucionales (CNN), diseñadas para el reconocimiento y análisis de imágenes y las redes neuronales recurrentes (RNN) ideales para el procesamiento de secuencias de datos como el texto o el audio [21].

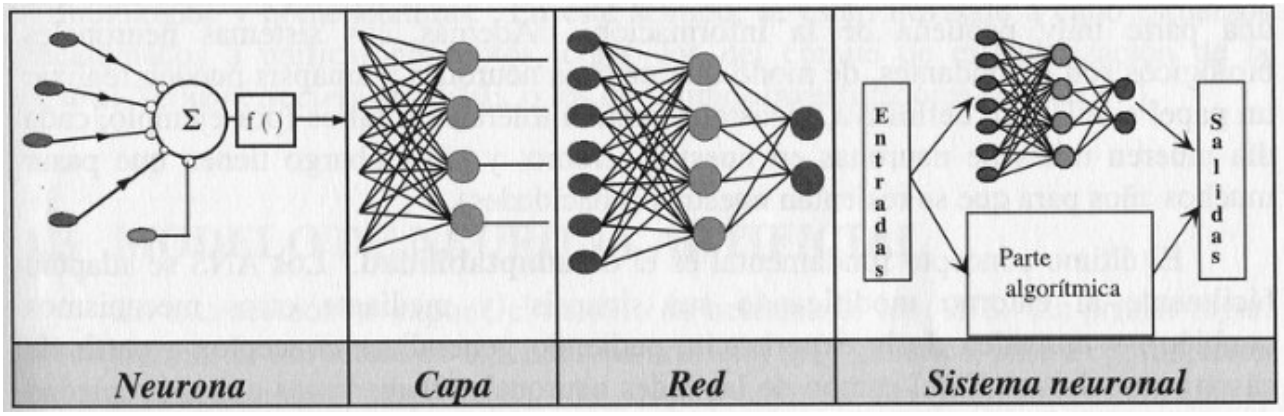

Figura 3. Sistema global de proceso de una red neuronal, [en línea] Disponible: https://www.researchgate.net/figure/Figura-1-Sistema-global-de-proceso-de-una-red-neuronal_fig1_268291232

El aprendizaje profundo ha revolucionado campos como el reconocimiento de voz, la visión por computadora y el procesamiento del lenguaje natural [22]. Estas aplicaciones logran que las máquinas realicen tareas que suelen ser naturales para los seres humanos, como identificar objetos en una imagen, interpretar el habla o traducir idiomas.

Su adopción masiva en la última década se ha visto impulsada por dos factores clave, por un lado la disponibilidad de grandes cantidades de datos y por el otro los avances en potencia computacional, impulsados por tecnologías como las unidades de procesamiento gráfico (GPU) y la computación en la nube reduciendo significativamente el tiempo de entrenamiento de los modelos.

Una característica fundamental de los algoritmos de aprendizaje profundo es que, en general, su rendimiento mejora a medida que se aumenta la cantidad de datos de entrenamiento. Esta capacidad de escalar en precisión y efectividad con mayores volúmenes de datos es una ventaja significativa sobre otros algoritmos de aprendizaje automático, que suelen enfrentar limitaciones de convergencia al trabajar con grandes datos [23]. La naturaleza del aprendizaje profundo, basada en su capacidad de aprender y mejorar a partir del procesamiento de datos complejos y no estructurados impulsa el desarrollo de la IA, abriendo nuevas posibilidades, especialmente en tareas de análisis de datos a gran escala y en aplicaciones que requieren un alto nivel de precisión y adaptabilidad [24].

2.4. Redes Neuronales Artificiales

Las redes neuronales artificiales (RNA) representan las estructuras fundamentales constitutivas del aprendizaje automático y aprendizaje profundo, inspiradas en el funcionamiento del cerebro humano. Las RNA se forman a partir de un conjunto de nodos interconectados, denominados neuronas artificiales, organizados en capas agrupados en un modelo, la primera se denomina capa de entrada, luego le siguen una o más capas intermedias u ocultas, finalizando con la última capa, denominada capa de salida. Cada neurona recibe el o los valores de una o varias variables, realiza un cálculo a partir de una función, denominada función de activación y entrega el resultado a la o las siguientes neuronas, permitiendo modelar relaciones complejas entre las variables de entrada y salida. Las RNA se entrenan mediante un proceso iterativo conocido como propagación hacia adelante y retropropagación. Durante la propagación hacia adelante, las entradas se transmiten a través de la red con el objetivo de calcular una predicción, posteriormente, se evalúa el error entre la predicción y la salida deseada utilizando una función, denominada función de pérdida. Este error se propaga hacia atrás a través de la red durante la retropropagación, ajustando los coeficientes o pesos de las conexiones entre las neuronas con el objetivo de reducir el error [25].

El uso de las redes neuronales artificiales ha demostrado ser efectivo en una amplia variedad de aplicaciones, como clasificación, regresión y detección de patrones [26]. En el contexto del phishing, las RNA se utilizan para analizar patrones en datos estructurados y no estructurados, permitiendo identificar elementos característicos de ataques maliciosos [27].

2.5. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN) son una extensión de las redes neuronales artificiales diseñadas específicamente para procesar datos con estructuras espaciales, como imágenes [28]. A diferencia de las estructuras tradicionales, las CNN aprovechan

operaciones de convolución para detectar patrones locales, como bordes, texturas y formas, que son cruciales en tareas de visión por computadora, esta característica las hace particularmente útiles en aplicaciones como reconocimiento facial, clasificación de imágenes y detección de objetos [29].

La arquitectura de las CNN incluye tres tipos principales de capas:

- **Capas de Convolución:** Estas capas aplican funciones, denominadas filtros o kernels sobre las imágenes de entrada con el objetivo de extraer características relevantes. Cada filtro responde a la identificación de un patrón específico, como bordes horizontales o verticales y produce un mapa de características que expone la presencia de dicho patrón en la imagen.
- **Capas de Pooling:** Reducen las dimensiones de los mapas de características mediante operaciones matemáticas, buscando conservar la información relevante a la vez que quita aquella no relevante, logrando disminuir la complejidad computacional y ayuda a evitar el sobreajuste, es decir, evitar que el modelo presente un buen desempeño para los datos de entrenamiento pero resulte en un bajo desempeño para los datos no conocidos.
- **Capas Completamente Conectadas:** Estas capas contienen conexiones entre todas sus neuronas, generalmente son las responsables de realizar la clasificación final basándose en las características o patrones extraídos, transfiriendo los resultados de la última capa oculta a la capa de salida.

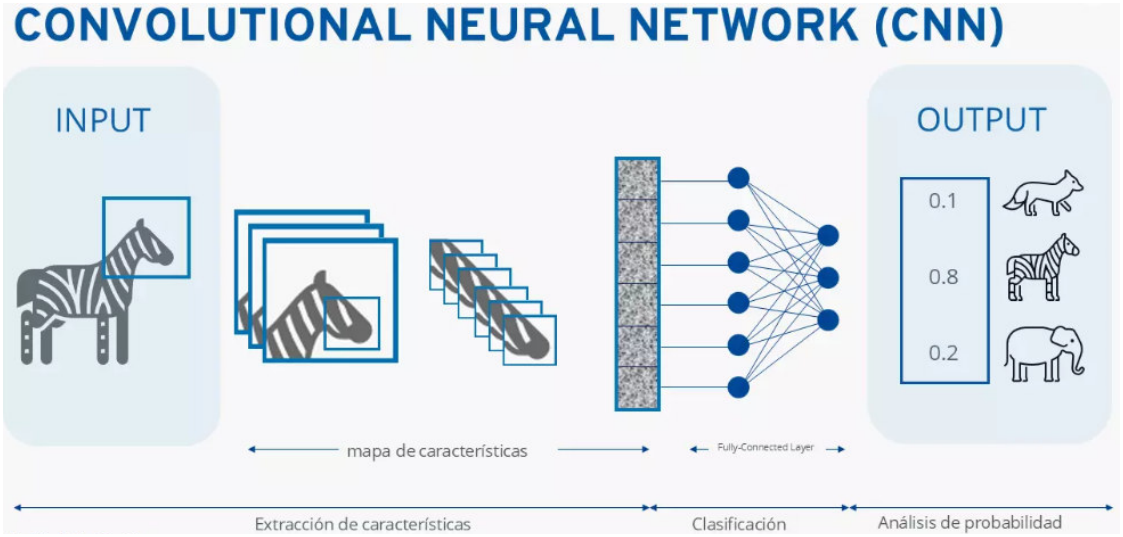

Figura 4. Arquitectura de las Redes Neuronales Convolucionales, [en línea] Disponible: https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/convolutional-neural-network/

Una de las ventajas de las redes neuronales convolucionales es su capacidad para aprender características jerárquicas. Las capas iniciales detectan patrones básicos, como bordes, mientras que las capas siguientes identifican características más complejas, como formas o incluso objetos completos. En el ámbito de la detección de phishing, las redes neuronales convolucionales se utilizan para extraer características de las imágenes correspondientes a páginas de los sitios web, identificando discrepancias visuales respecto de los sitios normales compatibles con las técnicas de suplantación de identidad [30].

2.6. Entrenamiento y ajuste de modelos

El entrenamiento y ajuste de modelos de aprendizaje profundo es un proceso iterativo y crítico con el objetivo de conseguir un buen desempeño en las predicciones de modelo. En este proceso se destacan los siguientes pasos y consideraciones:

2.6.2. Entrenamiento de modelos

El entrenamiento de los modelos de redes neuronales es un proceso fundamental en el aprendizaje profundo, que implica optimizar los pesos de la red de neuronas minimizando el error en las predicciones. Durante la etapa de entrenamiento se destacan las siguientes actividades:

Entrenamiento del modelo: se encarga de recibir un grupo de datos de entrada, denominado conjunto de entrenamiento, adaptados a las necesidades del modelo. Por ejemplo, en un modelo destinado a la clasificación, cada datos de entrenamiento debe tener su correspondiente etiqueta. El modelo realiza la predicción inicial de salida basado en los parámetros de inicialización, definidos por ejemplo, en forma aleatoria. Se compara la predicción del modelo con la etiqueta correcta utilizando la función de pérdida, por ejemplo error cuadrático medio para problemas de regresión y entropía cruzada para problemas de clasificación, como resultado se obtiene una cuantificación de la diferencia entre la predicción y la realidad. El error se propaga hacia atrás a través de las capas de la red, calculando el aporte de cada parámetro, pesos y sesgo, al error total. Luego se utilizan algoritmos de optimización, como el descenso del gradiente estocástico (SGD) o sus variantes, por ejemplo Adam, para ajustar los parámetros de la red en la dirección que minimice la función de pérdida. Durante el entrenamiento se ajusta la tasa de aprendizaje, definiendo el paso que controla la velocidad a la que se actualizan los parámetros y en consecuencia la convergencia del modelo. Los pasos anteriores se repiten durante un cierto número de iteraciones, denominadas épocas, hasta que el modelo alcanza el nivel de precisión definido o se cumple un criterio de parada, por ejemplo, si las métricas no mejoran durante cinco iteraciones.

Validación del modelo: Luego del entrenamiento se evalúa el comportamiento del modelo con datos que no conoce, generalmente denominado conjunto de test. Se calculan las métricas definidas, por ejemplo, precisión o sensibilidad para tareas de clasificación y error cuadrático medio para modelos enfocados en resolver problemas de regresión. Las métricas permiten medir la capacidad de generalización del modelo. La etapa de validación permite identificar problemas como el sobreajuste, cuando el modelo responde bien a los datos de entrenamiento pero no tiene un buen desempeño con los datos de

test, a la vez que permite identificar problemas de subajuste, cuando el modelo no logra capturar los patrones en el conjunto de datos durante el entrenamiento.

En la etapa de entrenamiento y validación del modelo, se destaca la configuración de los denominados hiperparámetros, por ejemplo, la tasa de aprendizaje y el número de épocas. Este ajuste se puede realizar en forma manual, evaluando el resultado luego de cada iteración o encargar la tarea a algoritmos especializados, como la búsqueda aleatoria, búsqueda en cuadrícula o enfoques más avanzados, como la optimización bayesiana.

En el contexto del phishing, un modelo bien entrenado y ajustado permite identificar elementos sospechosos en interfaces visuales. Además, la elección de un conjunto de datos representativos y bien equilibrados ayuda a garantizar que el modelo generalice adecuadamente para patrones compatibles con variaciones de los ataques de phishing [31].

2.7. Ciberseguridad

La ciberseguridad es el conjunto de prácticas, procesos y tecnologías orientadas a proteger la integridad, confidencialidad y disponibilidad de sistemas, redes y datos frente a ataques y accesos no autorizados [32]. A medida que la digitalización se ha expandido, conectando prácticamente todos los aspectos de la vida cotidiana y empresarial, la ciberseguridad ha pasado de ser una simple medida de protección a ser un requisito fundamental en la sociedad moderna, abarcando múltiples capas de protección distribuidas a través de dispositivos y redes que incluyen desde el software hasta las estrategias organizativas de gestión de riesgos conformando un problema socio-técnico [33].

Las amenazas convergen desde distintos vectores materializando y técnicas, desde ataques automatizados hasta tácticas de ingeniería social, donde el phishing se destaca como una de las técnicas más utilizadas [34]. A medida que las técnicas de ataque se vuelven más complejas, la ciberseguridad también se vuelve más crítica y dinámica, requiriendo una vigilancia constante y la actualización continua de las herramientas de

defensa, en este campo la inteligencia artificial junto a la ciencia de datos pueden aportar algoritmos inteligentes para los sistemas de detección de intrusiones y cortafuegos [35].

2.8. Suplantación de identidad (phishing)

El phishing, o suplantación de identidad, es una técnica de ataque informático que se basa en el uso de métodos de ingeniería social para engañar a los usuarios y hacer que compartan información sensible, como contraseñas, datos bancarios o credenciales de acceso. Los atacantes emplean una variedad de tácticas para hacerse pasar por entidades legítimas, tales como bancos, instituciones gubernamentales o incluso redes sociales, utilizando como vector para llegar a la víctima un correo electrónico, mensajes de texto o páginas web falsificadas que imitan sitios confiables [36]. Este engaño tiene como objetivo que los usuarios revelen información confidencial o realicen acciones que comprometan la seguridad de sus datos y sistemas.

Las técnicas de phishing se mantienen en la cumbre de los rankings de incidentes de seguridad, con la combinación de elementos visuales auténticos, junto a un lenguaje persuasivo, siguen siendo efectivas. Además, los avances tecnológicos han permitido enfocar el ataque hacia una persona en particular, dando lugar a nuevos métodos, como el spear phishing, que es un ataque dirigido a individuos específicos y el whaling, dirigido a altos ejecutivos [37].

2.9. Aprendizaje profundo y Phishing

En los últimos años, el uso de técnicas de aprendizaje profundo han sumado relevancia en el ámbito de la ciberseguridad, entre ellas, destacamos las relacionadas con la detección del phishing debido al éxito y alto impacto de estos ataques. El phishing, al simular interfaces de usuario legítimas, tanto en correos electrónicos como en sitios web, plantea un desafío para los métodos de detección de amenazas tradicionales. Sin embargo, el aprendizaje profundo, gracias a su capacidad para procesar grandes volúmenes de datos y extraer patrones complejos de manera autónoma, ha logrado identificar características

sutiles en imágenes y elementos visuales compatibles con las técnicas de suplantación [38, 39].

Los modelos de Deep Learning pueden analizar minuciosamente los aspectos gráficos y de diseño en interfaces falsas, reconociendo patrones que el ojo humano o los algoritmos convencionales podrían pasar por alto. Esta capacidad de "ver" en profundidad permite que las herramientas de detección de aprendizaje profundo identifiquen instancias de phishing de manera más precisa y rápida, mejorando la capacidad de detección temprana frente a estos ataques. La sinergia entre aprendizaje profundo y ciberseguridad mejora la capacidad de detección de los sistemas de seguridad informática colaborando con la prevención de ataques de phishing apoyados en algoritmos capaces de adaptarse a las tácticas de los atacantes [40].

3. Estado del Arte

3.1. Introducción al Estado del Arte

En los siguientes párrafos se abordará la detección de patrones de phishing mediante técnicas de aprendizaje profundo, identificando las estrategias, los avances tecnológicos y las áreas de oportunidad. El phishing, es una de las formas más comunes y efectivas de ataque informático, caracterizado por su capacidad de adaptación y evolución, por lo cual demanda nuevas soluciones tecnológicas para combatirlo de manera eficiente. Por su lado, el área de visión artificial y las redes neuronales convolucionales (CNN) han demostrado ser herramientas útiles para la identificación automática de patrones en imágenes, ofreciendo un potencial abordaje para la detección temprana de ataques compatibles con phishing [41] .

3.2. Análisis de las regiones de interés en Ciberseguridad

Las regiones de interés o ROIs constituyen áreas específicas en una imagen, las cuales poseen información crítica para el análisis del problema, donde las técnicas de segmentación no supervisadas mediante redes neuronales convolucionales han colaborado en la construcción de algoritmos robustos. En el ámbito del phishing, las ROIs pueden incluir elementos clave como logos de marcas adulterados, diseños de interfaces sospechosos y patrones gráficos que imitan sitios web confiables.

Figura 5. Identificación de regiones de interés, [en línea] Disponible: https://preyproject.com/es/blog/si-no-sabes-que-es-el-phishing-estas-en-problemas

La segmentación de ROIs en phishing comparte principios comunes con la detección de tumores en radiografías o la identificación de anomalías en estudios biomédicos [42]. Ambas disciplinas se benefician del uso de arquitecturas basadas en redes neuronales convolucionales afinadas para localizar y clasificar objetos en imágenes con alta precisión.

3.3. Evolución de las técnicas de detección de phishing

Los métodos tradicionales de detección de phishing se basaban principalmente en sistemas de listas negras y algoritmos basados en reglas. Estas técnicas, aunque efectivas en su momento, demostraron ser insuficientes ante la creciente sofisticación de los ataques. Posteriormente, el aprendizaje automático permitió mejorar las técnicas de detección al dotar a los sistemas de la capacidad de aprendizaje a partir de datos etiquetados, y posteriormente identificar patrones compatibles en un un texto, por ejemplo en los correos electrónicos. En la última etapa, se incorporaron los algoritmos que permiten extraer patrones de las imágenes, incorporando la capacidad de analizar sitios web completos, abriendo el camino para la explotación del aprendizaje profundo [43].

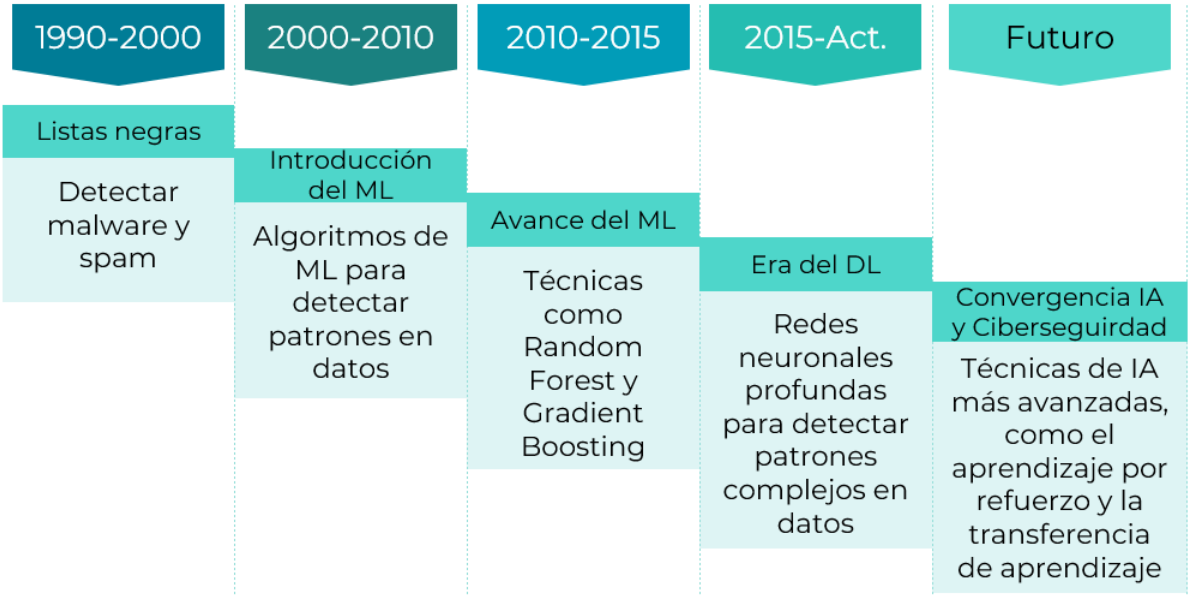

Figura 6. Detección de phishing en el tiempo. Elaboración propia.

El aprendizaje profundo introduce una nueva perspectiva al abordar la detección de phishing mediante técnicas especializadas en el análisis de datos no estructurados, como las imágenes; en este área se destacan las redes neuronales convolucionales, mejorando la precisión de las herramientas de ciberseguridad al incorporar algoritmos con la capacidad de generalizar, es decir, detectar en imágenes nuevas patrones compatibles con el problema para el cual están entrenados [44].

3.4. Revisión de estudios relevantes

Se realizó la exploración en el catálogo OpenAlex [44], y otros como Google Scholar y Microsoft Academic, donde encontramos resultados alentadores en el uso de los algoritmos de aprendizaje automático aplicados a la ciberseguridad en general y en la detección de phishing en particular, basados en la capacidad de las redes neuronales convolucionales para identificar patrones maliciosos en los correos electrónicos y sitios web, permitiendo clasificar entre interfaces genuinas y fraudulentas [45, 46, 47].

Se detecta la búsqueda de mejoras en el desempeño a partir del diseño de modelos basados en arquitecturas híbridas, combinando las capacidades de las redes neuronales

convolucionales, con redes neuronales recurrentes y redes adversariales, entre otras, dotando de capacidades para analizar tanto las imágenes como el texto de las entradas, mostrando mejoras en el desempeño al integrar información proveniente de distintas fuentes [48, 49, 50].

En todos los casos, se resalta la importancia de los conjuntos de datos en el desempeño de los modelos de aprendizaje profundo. La calidad y representatividad del conjunto de datos se considera un aspecto clave para analizar la capacidad de generalización del modelo sobre nuevas instancias [51, 52].

3.5. Aplicaciones específicas de aprendizaje profundo al phishing

El aprendizaje profundo ha demostrado resultados satisfactorios en la detección de regiones de interés compatibles con phishing en las interfaces de usuario de los sitios web y correos electrónicos, utilizados como medio para engañar a las víctimas [53]. En particular, las redes neuronales convolucionales han demostrado resultados alentadores en la clasificación de sitios fraudulentos a partir de la extracción de patrones en las imágenes compatibles con elementos fraudulentos, como logotipos falsificados, esquemas de colores sospechosos y estructuras de diseño inconsistentes [54]. Un sistema basado en redes neuronales convolucionales tiene la capacidad de detectar discrepancias entre un logotipo utilizado en un correo electrónico y el original de la organización que supuestamente representa. Estas capacidades permiten no solo identificar amenazas, sino también actuar como una herramienta preventiva para alertar a los usuarios antes de que interactúen con contenido potencialmente peligroso [55].

En los trabajos analizados se destacan los abordajes basados en la combinación de las técnicas de aprendizaje profundo con técnicas de procesamiento de lenguaje natural [56] permitiendo analizar tanto el contenido textual como las imágenes, mejorando la capacidad de detección. Las soluciones integradas representan el estado del arte en la lucha contra el phishing, proporcionando sistemas más robustos y adaptativos frente a las tácticas en constante evolución de los atacantes [57, 58].

4. Implementación

A continuación se introducen las herramientas y el lenguaje de programación seleccionado, luego se destacan las características del modelo base utilizado y por último se detalla el conjunto de datos utilizado para el entrenamiento y validación del modelo.

4.1. Herramientas

Google Collaboratory

La ejecución del código y fundamentalmente las actividades de entrenamiento y validación del modelo se realizaron en la plataforma Google Colaboratory o Google Colab (Google Research, s.f.) [59]. Es un servicio dispuesto en la nube que ofrece acceso a recursos de cómputo como GPUs (unidades de procesamiento gráfico) y TPUs (unidades de procesamiento tensorial), especialmente requeridas en tareas intensivas en cómputo como los modelos de aprendizaje profundo.

Lenguaje de Programación python

El análisis de los datos, diseño, entrenamiento y validación del modelo y presentación de los resultados se codificó en el lenguaje python [60], seleccionando bibliotecas y frameworks especializados para ciencia de datos y aprendizaje automático para acelerar la investigación y desarrollo. A continuación citamos las bibliotecas utilizadas:

- TensorFlow [61] / Keras [62]**:** se utilizaron para diseñar, entrenar y evaluar los modelos de aprendizaje profundo.
- NumPy [63] y Pandas [64]**:** permitieron facilitar la manipulación y el análisis del conjunto de datos.
- Matplotlib [65]: permitió simplificar la visualización gráfica de los resultados obtenidos durante el análisis y entrenamiento de los modelos.
- Sklearn [66]: facilitó las actividades vinculadas al preprocesamiento de datos y la evaluación de modelos mediante métricas estándar como precisión, recall y F1-score.

4.2. Modelo VGG16

El modelo VGG16 [67], desarrollado por el Visual Geometry Group (VGG) de la Universidad de Oxford, es una arquitectura de red neuronal convolucional ampliamente utilizada en el campo de la visión por computadora, especialmente para tareas de clasificación de imágenes. Destaca por su diseño simple y efectivo y las siguientes características clave:

- Profundidad: el nombre VGG16 hace referencia a las 16 capas de procesamiento (capas convolucionales y totalmente conectadas) que conforman la red. Su profundidad le permite aprender características visuales complejas.
- Uniformidad: utiliza solamente filtros convolucionales de tamaño 3x3 en todas sus capas convolucionales, logrando simplificar la arquitectura y facilitar su entrenamiento.
- Max Pooling: incorpora capas de max pooling para reducir la dimensionalidad de las representaciones intermedias y mejorar la robustez frente a pequeñas variaciones en las características visuales.
- Preentrenamiento en ImageNet: el modelo fue preentrenado en el conjunto de datos ImageNet [68], el cual incluye millones de imágenes etiquetadas. Este preentrenamiento le permite captar características visuales de bajo nivel útiles para una amplia gama de tareas de visión por computadora.
- Aprendizaje por transferencia o Transfer Learning: el modelo VGG16 se puede emplear como base para resolver problemas específicos aplicando la técnica de aprendizaje por transferencia, requiriendo una menor cantidad de datos, tiempo y recursos computacionales para el entrenamiento y ajuste al nuevo problema.

El modelo VGG16 se ha posicionado como una herramienta de utilidad en la resolución de situaciones problemáticas donde se requiera clasificación y segmentación de imágenes o localización y detección de objetos, presentándose como un complemento al abordaje de los desafíos complejos y dinámicos presentes en la lucha contra las amenazas digitales [69].

4.3. Conjunto de datos o dataset

4.3.1. Descripción del conjunto de datos

En el trabajo se utilizó una variación del conjunto de datos Phish-IRIS [70] desarrollado por el grupo HUMIR Lab de la Universidad Hacettepe de Ankara para investigaciones académicas sobre detección de phishing mediante sistemas de visión artificial. El conjunto de datos original incluye dos grupos de imágenes, uno destinado al entrenamiento, formado por 1313 imágenes y otro para validación, formado por 1539 imágenes. Cada conjunto se organiza en 15 clases, 1 de ellas, denominada *other* contiene imágenes de sitios normales o sin phishing y las 14 clases restantes corresponden a imágenes de sitios afectados por phishing agrupados por la marca o empresa a la cual corresponde.

Las imágenes se acomodan en carpetas cuyo nombre representa el conjunto de datos al cual pertenecen, así se dispone una carpeta denominada *train* y otra denominada *test*, donde se ubican las clases que contienen las imágenes destinadas al entrenamiento y test .

data/

├── *train/*

  ├── *phishing/* *#14 clases (399 imágenes)*

  └── *normal/* *#1 clase (910 imágenes)*

└── *test/*

  ├── *phishing/* *#14 clases (536 imágenes)*

  └── *normal/* *#1 clase (998 imágenes)*

Figura 7. Estructura original del conjunto de datos. Elaboración propia.

El análisis inicial del conjunto original permite identificar que su estructura no resultaba óptima para el objetivo del trabajo, además, de presentar un desbalance en las clases. La división de clases en marcas no aportaba valor, por lo cual se tomó la decisión de reorganizar el conjunto de datos con dos categorías, a la vez que se procuró balancear las clases. Se decide configurar la categoría *normal* para agrupar las imágenes correspondientes a sitios sin phishing y la categoría *phishing* para contener las imágenes de los sitios con phishing, conformando cada grupo con 1300 imágenes seleccionadas en forma aleatoria del conjunto de datos original. Los hallazgos y propuestas anteriores resultaron en la reorganización de las carpetas e imágenes dentro de la carpeta *data, así*, la nueva estructura requiere la creación de las carpetas *phishing* y *normal* y el reacomodamiento en la carpeta phishing de 1.300 imágenes seleccionadas de las 14 clases correspondientes a las marcas tomadas en forma aleatoria de las carpetas *train* y *test* y luego, se seleccionan 1.300 imágenes en forma aleatoria del grupo *train* y *test* de la clase *other*.

data/

├── phishing/ #1300 imágenes

└── normal/ #1300 imágenes

Figura 8. Reorganización del conjunto de datos en dos clases. Elaboración propia.

Durante la etapa de ajuste inicial, se definió la necesidad de probar el modelo en un conjunto de datos acotado, y posteriormente a conocer el tiempo requerido para las etapas de entrenamiento y validación avanzar al conjunto de datos objetivo formado por 1.300 imágenes de phishing y 1.300 imágenes de sitios normales. Como abordaje del problema se consideró conveniente separar el conjunto de datos original en tres conjuntos de datos, surgiendo así el *dataset40*, conteniendo 40 imágenes de phishing y 40 imágenes de sitios normales, luego se configura el *dataset500*, con 500 imágenes en cada

clase, y por último se considera el *dataset1300* conteniendo 1.300 imágenes para la categoría phishing y 1.300 para la categoría normal. En resumen se trabajará con los siguientes conjuntos de datos:

- **Dataset1300:** formado por 1.300 imágenes en la clase phishing y 1.300 imágenes en la clase normal.
- **Dataset500:** incorpora 500 imágenes para la clase phishing y 500 imágenes para la clase normal, utilizado para evaluar el tiempo de procesamiento en la infraestructura disponible y el desempeño del modelo.
- **Dataset40:** contiene 40 imágenes en la clase phishing y 40 imágenes en la clase normal, diseñado para pruebas exploratorias iniciales.

4.3.2. Carga y preparación del conjunto de datos

En las siguientes líneas se describen las tareas requeridas para organizar y preparar los datos para su procesamiento presentadas a través de su implementación en python.

Importar el dataset:

```
from google.colab import files

# Subir archivos

uploaded = files.upload()
```

```
# Instalar la herramienta para descomprimir

!apt-get install unrar

# Descomprimir el archivo

!unrar x dataset1300
```

Establecer la ubicación de los datos:

```
# Directorio principal de los datos

data_path = '/content/dataset1300/data'
```

Durante la preparación del conjunto de datos se deben organizar las imágenes respetando la estructura requerida por la librería encargada del procesamiento de datos y el entrenamiento del modelo. Esta librería requiere que las imágenes se acomoden en carpetas, cuyo nombre será representativo de la clase a la cual pertenece, en nuestro caso como el modelo es un clasificador binario, tenemos dos clases, correspondientes a las imágenes de sitio normales y sitios con phishing. Al definir la estructura de carpetas anterior, junto con el acomodamiento de las imágenes en cada carpeta estamos cumpliendo los requerimientos de la librería definida para realizar etiquetado, entrenamiento y validación del modelo.

```
# Obtener todas las imágenes y sus etiquetas

image_paths = []

labels = []

for class_name in ['normal', 'phishing']:

    class_path = os.path.join(data_path, class_name)

    for extension in ['*.jpg', '*.png']:

            for  image_path  in  glob.glob(os.path.join(class_path,
extension)):
```

```
            image_paths.append(image_path)

            labels.append(class_name)
```

4.3.3.Preparación y preprocesamiento de datos

Con el objetivo de facilitar la manipulación y procesamiento de los datos se acomodan las rutas de las imágenes y sus etiquetas en una estructura de datos del tipo dataframe utilizando la biblioteca *pandas*.

```
# Crear un DataFrame con las rutas de las imágenes y las etiquetas
data = pd.DataFrame({'image_path': image_paths, 'label': labels})
```

A continuación se crean las carpetas *train* y *test* para almacenar las imágenes correspondientes a los conjuntos de datos destinados al entrenamiento y prueba respectivamente, dentro de cada grupo se generan las carpetas para la categoría phishing y normal. La organización de carpetas anteriormente detallada cumple con los requerimientos de la librería encargada de gestionar y transformar los datos durante las etapas de entrenamiento y validación del modelo.

```
# Crear directorios temporales
train_dir = 'data/train'
test_dir = 'data/test'
os.makedirs(train_dir, exist_ok=True)
```

```
os.makedirs(test_dir, exist_ok=True)
```

Posteriormente, se utiliza la función `train_test_split` de la biblioteca scikit-learn para dividir el conjunto de datos en entrenamiento y prueba, asignando un 80% y 20% de las imágenes respectivamente y asegurando una distribución equilibrada en ambas clases mediante la opción stratify.

```
# Dividir en entrenamiento y test (80/20)

train_df, test_df = train_test_split(

    data,

    test_size=0.2,

    random_state=42,

      stratify=data['label'] # Garantiza la misma distribución en
las clases

)
```

Luego se utiliza una función personalizada para repartir las imágenes de cada grupo (entrenamiento y test) en las carpetas *data/train/phishing* y *data/train/normal* que les corresponden de acuerdo a su etiqueta, para lo cual se utiliza la información almacenada en el dataframe *data* generado anteriormente. Cómo resultado logramos acomodar el conjunto de datos de acuerdo a los requerimientos de la librería seleccionada para realizar el entrenamiento y evaluación del modelo, además, la separación física de las imágenes refuerza la integridad del proceso, minimizando los errores derivados de la manipulación de los datos entre los conjuntos.

```
# Crear carpetas para cada clase dentro de train y test

for folder in ['train', 'test']:

    for class_name in ['normal', 'phishing']:
```

```
            os.makedirs(os.path.join(f'data/{folder}', class_name),
exist_ok=True)


# Definir una función para copiar las imágenes de una carpeta en
otra según su etiqueta

def copy_images(df, target_dir):

    for _, row in df.iterrows():

        src_path = row['image_path']

        class_name = row['label']

                dst_path = os.path.join(target_dir, class_name,
os.path.basename(src_path))

        shutil.copy(src_path, dst_path)

# Copiar imágenes de entrenamiento separadas por train_test_split

copy_images(train_df, 'data/train')

# Copiar imágenes de test separadas por train_test_split

copy_images(test_df, 'data/test')
```

4.4. Construcción, ajuste y preparación del modelo

En las siguientes líneas se describen las actividades llevadas a cabo para diseñar un clasificador binario a partir del modelo VGG16 propuesto por Simonyan y Zisserman, pre-entrenado en el conjunto de datos *ImageNet* [71], donde se han reemplazado las últimas capas del modelo original por dos capas densas personalizadas. Además, las

primeras capas convolucionales se congelaron para preservar las características generales aprendidas en ImageNet. Como resultado se obtendrá un modelo configurado para la tarea de clasificación binaria pre-entrenado en la identificación de patrones en imágenes. Posteriormente, se entrenará este modelo en el reconocimiento de patrones compatibles con phishing.

Este proceso se reconoce como transferencia de aprendizaje o transfer learning, como resultado consigue reducir el esfuerzo computacional y la cantidad de datos necesarios para lograr un modelo con un desempeño aceptable, logrado a través de la reutilización del conocimiento aprendido en el entrenamiento sobre un conjunto de datos extenso donde el modelo aprendió patrones generales. Luego, este modelo generalista debe ser entrenado y validado con datos representativos del problema a resolver.

Diseño y compilación del clasificador binario

En primer lugar se carga el modelo VGG16 pre-entrenado, luego se añaden y configuran las capas densas para realizar la clasificación binaria.

```
# Cargar el modelo VGG16 pre-entrenado sin la capa de clasificación superior

base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

Dado que el problema a resolver requiere un clasificador binario, es necesario incorporar capas densas al modelo enfocadas en esta tarea. Entonces, las capas del modelo base permiten extraer características complejas y significativas de las imágenes y las capas densas actúan como un clasificador basado en las características que encuentre en el dato de entrada compatibles con el dominio del problema para el cual el modelo está entrenado, en nuestro caso el modelo analizará la imagen de entrada buscando características compatibles con phishing y como resultado entregará un peso o

probabilidad para la clase phishing y normal calculado en base a los patrones identificados en la imagen.

Las siguientes líneas de código configuran la arquitectura del modelo pre-entrenado para realizar una tarea de clasificación binaria.

```
# Agregar capas densas para la clasificación binaria

x = Flatten()(base_model.output)

predictions = Dense(1, activation='sigmoid')(x)

# Crear el modelo según las configuraciones anteriormente aplicadas

model = Model(inputs=base_model.input, outputs=predictions)
```

Luego se compila el modelo diseñado, definiendo el optimizador, la función de pérdida que permite cuantificar la distancia entre las predicciones del modelo y las etiquetas verdaderas y las métricas para evaluar el desempeño en las etapas de entrenamiento y validación. En nuestro caso, decidimos utilizar Adam, un optimizador adaptativo y eficiente en una amplia variedad de tareas de aprendizaje profundo [72]. Como función de pérdida seleccionamos la entropía cruzada binaria, de amplio uso en problemas de clasificación binaria [73], y finalmente para evaluar el desempeño del modelo, seleccionamos las métricas accuracy, precision y recall, las cuales son ampliamente empleadas para evaluar el modelo en términos de aciertos, falsos positivos y falsos negativos, respectivamente [74].

En cuanto a las métricas, debemos mencionar que consideramos una buena práctica considerar el dominio del problema al momento de analizar cuáles métricas utilizar, por ejemplo, en nuestro caso, estamos abordando un problemas en el campo de la ciberseguridad, donde los errores de clasificación del modelo tienen consecuencias en las actividades de las personas, por este motivo consideramos que se debe prestar especial atención al desempeño del modelo en relación a los falsos positivos (cuando el modelo

clasifica como phishing una imagen que en realidad corresponde a un sitio normal) y los falsos negativos (cuando el modelo clasifica como normal una imagen que es phishing). Luego, en la evaluación del desempeño, el analista definirá si el costo de un falso positivo es alto priorizará la precisión, por otro lado, si el costo de un falso negativo es alto debería prestar atención al recall.

```
# Compilar el modelo

model.compile(optimizer='adam',

              loss='binary_crossentropy',

              metrics=[

                  BinaryAccuracy(name='accuracy'),

                  Precision(name='precision'),

                  Recall(name='recall')

              ])
```

Procesamiento de los datos de entrada

La siguiente etapa, se concentra en ajustar los datos de entrada a los requerimientos del modelo base VGG16, y simultáneamente, aplicar transformaciones cuyo objetivo es mejorar la capacidad de generalización, para ello utilizaremos *ImageDataGenerator* de Keras, la cual nos permite generar lotes de imágenes, a la vez que permite aplicar de forma aleatoria transformaciones como rotación y escalado. Esta técnica, conocida como aumento de datos, presenta al modelo una mayor diversidad de imágenes durante el entrenamiento. A continuación presentamos las transformaciones aplicadas:

- **Redimensionamiento:** las imágenes se ajustaron a 224x224 píxeles para cumplir con los requisitos del modelo VGG16.

- **Normalización:** los valores de los píxeles se escalaron al rango [0,1].
- **Aumento de datos:** se aplicaron técnicas de zoom, rotación, traslación y volteo horizontal.

Las transformaciones buscan reducir el riesgo de sobreajuste (*overfitting),* situación que se presenta cuando el modelo se adapta demasiado a los datos de entrenamiento, perdiendo capacidad de generalización. En el contexto de nuestro problema, el aumento de datos tiene el objetivo de representar características compatibles con las imágenes provenientes de capturas de pantalla.

```
# Cargar la librería para el procesamiento de datos

from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
# Crear  generadores de datos

train_datagen = ImageDataGenerator(

    rescale=1./255, # Normalizar los píxeles

    rotation_range=20, # Rotación aleatoria

    width_shift_range=0.2, # Desplazamiento horizontal

    height_shift_range=0.2, # Desplazamiento vertical

    shear_range=0.2, # Deformación por shear (cizalladura)

    zoom_range=0.2, # Zoom aleatorio

    horizontal_flip=True, # Volteo horizontal

    fill_mode='nearest',
```

```
    validation_split=0.2
)
test_datagen =ImageDataGenerator(
    rescale=1./255
) # En el conjunto de test solamente normalizar
```

Pensando en la optimización de tiempo y recursos computacionales requeridos en el entrenamiento del modelo, se utilizan los generadores de datos, provistos por la librería *ImageDataGenerator*. Estos generadores se encargan de dividir el conjunto de datos en lotes de tamaño específico (definidos en el parámetro batch_size), suministrando al modelo las imágenes preprocesadas de forma secuencial. Además, los generadores gestionan en forma eficiente la selección de los datos para las etapas de entrenamiento, validación y prueba, garantizando las condiciones de independencia a la vez que facilitan la gestión eficiente de los datos.

```
# Crear los generadores
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224), # Ajusta el tamaño al modelo VGG16
    batch_size=5,
    class_mode='binary',
    subset='training',
```

```
    seed=42,
    shuffle= True # Mezcla los datos en cada época
)
validation_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224,224),
    batch_size=5,
    class_mode='binary',
    subset='validation',
    seed=42,
    shuffle=True
)
```

```
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224,224),
    batch_size= 3,
    class_mode='binary',
    seed=42,
    shuffle=False
```

```
)
```

4.5. Entrenamiento del modelo

El entrenamiento del modelo se apoya en los generadores de datos, los cuales se encargan de llevar adelante la actividad alimentando el modelo con lotes de imágenes ajustados a las transformaciones definidas. En el proceso se utiliza la función *model.fit* integrada en el paquete Scikit-learn [75], entre otras facilidades ofrece la posibilidad de configurar los atributos denominadas hiperparámetros, en forma manual o entregando la búsqueda de la mejor combinación a un algoritmo, por ejemplo GridSearchCV [76] del paquete Scikit-learn. Considerando la limitación de los recursos computacionales disponibles se consideró conveniente configurar los hiperparámetros en forma manual. En primer término existe la posibilidad de definir la cantidad de épocas o pasadas completas del algoritmo en el conjunto de entrenamiento, y además, podemos definir si deseamos que se monitoree el proceso y se detenga en caso que las métricas no mejoren luego de una determinada cantidad de pasadas. También podemos definir la cantidad de pasos o lotes a utilizar en cada época. A continuación presentamos los hiperparámetros configurados y los valores asignados:

- *epochs*=32: especifica un máximo de 32 épocas para el entrenamiento. Una época corresponde a una iteración o recorrido a través del conjunto de datos de entrenamiento.
- *Earlystopping*: si la precisión (accuracy) no mejora durante 5 épocas consecutivas (patience=5), se detiene el entrenamiento.
- *steps_per_epoch*=len(train_generator): define el número de pasos o lotes por época para el entrenamiento. En nuestro caso lo configuramos como el número total de lotes asignados por el generador para la etapa de entrenamiento.
- *validation_steps*=len(validation_generator): indica el número de pasos para la etapa de validación. En este caso lo configuramos como la cantidad de lotes en el generador definido para la etapa de validación.

```
# Entrenar el modelo
history = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=32,
    validation_data=validation_generator,
    validation_steps=len(validation_generator),
    callbacks=[EarlyStopping(monitor='val_accuracy', patience=5)],
        verbose=1  # Imprime información sobre el progreso del entrenamiento
)
```

Con el objetivo de facilitar el seguimiento de la evolución de los parámetros se decide almacenar los valores en la variable *history*, conteniendo información valiosa para analizar la capacidad de generalización del modelo, a la vez que ayuda a detectar problemas como el sobreajuste, presente cuando el modelo muestra un buen desempeño en el entrenamiento pero presenta un desempeño pobre al enfrentarse con nuevos datos, sería como si el algoritmo hubiera memorizado los patrones presentes en las imágenes de entrenamiento en lugar de aprender las relaciones subyacentes en las regiones de interés de las imágenes); la información almacenada también ayuda a detectar las causas del subajuste, cuando el modelo no ha logrado detectar y aprender las características necesarias de las imágenes para realizar buenas predicciones.

```
# Acceder a la pérdida (loss) y exactitud (accuracy) en el conjunto de entrenamiento

loss = history.history['loss']

accuracy = history.history['accuracy']

# Graficar la pérdida y accuracy

plt.plot(loss, label='loss')

plt.plot(accuracy, label='accuracy')

plt.xlabel('Época')

plt.ylabel('Valor')

plt.legend()

plt.show()
```

Recordando, las métricas que definimos utilizar para evaluar el desempeño del modelo, son la *pérdida o loss*, que representa una medida cuantitativa del ajuste del modelo a los datos, en este caso sería deseable que el valor de la pérdida sea bajo, indicando que las predicciones del modelo están cerca de los valores reales; completamos el análisis con la medida de *exactitud o accuracy*, la cual mide la proporción de predicciones correctas realizadas por el modelo, en este caso una exactitud alta indica que el modelo está clasificando correctamente los datos. Observando la evolución de los valores de las métricas, podemos detectar indicios de *sobreajuste*, cuando la pérdida en el conjunto de entrenamiento disminuye mientras que la pérdida en el conjunto de validación aumenta y se estabiliza en un valor alto, o también puede ser un indicador de sobreajuste cuando los valores de exactitud para el entrenamiento y validación difieren en una gran magnitud; complementa lo anterior, el análisis respecto al *subajuste*, observable cuando los valores de la pérdida se estabilizan en valores altos, por otro lado, un *buen ajuste* se observa

cuando la pérdida disminuye mientras que la exactitud aumenta y se estabiliza en valores altos.

Basados en la exactitud y pérdida, se analizan los resultados en los tres conjuntos de datos, *dataset40*, *dataset500* y *dataset1300*. En la prueba inicial el modelo fue entrenado con el dataset40, iterando durante 10 épocas. Los resultados obtenidos presentan oscilaciones tanto en la pérdida (loss) como en la exactitud (accuracy), alternando entre valores altos y bajos sin una tendencia clara. Una potencial causa de la falta de convergencia puede ser que el número reducido de ejemplos no sea suficiente para que el modelo logre converger y adquiera la capacidad de generalización.

En la siguiente prueba, se entrena el modelo con el dataset500, logrando iterar sobre 22 épocas. Si bien, aún se observan oscilaciones tanto en la pérdida (loss) como en la exactitud (accuracy), los valores son menos pronunciados que los obtenidos en la prueba con el dataset40. Los resultados permiten inferir que si bien el dataset de tamaño 500 proporciona mayor diversidad y representatividad, brindando mayor solidez al aprendizaje, todavía es insuficiente para lograr una convergencia adecuada.

Finalmente, se realiza el entrenamiento con el dataset1300 iterando durante 22 épocas. Los resultados mejoran respecto a las oscilaciones de las pruebas anteriores. La exactitud (accuracy) alcanza picos más altos y con mayor frecuencia, reflejando una mejora en la capacidad del modelo para generalizar, no obstante, la pérdida (loss) no logra converger en valores bajos, por lo cual, si bien el incremento del tamaño del dataset contribuye a mejorar los resultados, aún resulta insuficiente para obtener un modelo robusto.

En resumen, podemos resaltar los siguiente resultados del entrenamiento del modelo:

- **Dataset40:** su reducido tamaño no permite que el modelo generalice ni aprenda correctamente, observando gran variabilidad y falta de convergencia en las métricas.
- **Dataset500:** presenta mejores resultados que el dataset40, pero las oscilaciones en las métricas persisten, reflejando que el modelo aún tiene problemas para converger.

- **Dataset1300:** proporciona mejores resultados que los casos anteriores, mostrando picos de precisión más altos y menor variabilidad en las métricas. Sin embargo, el modelo aún presenta oscilaciones, indicando que aún no se ha logrado un modelo sólido.

La etapa de entrenamiento presenta mejoras a partir del incremento del conjunto de datos pero se requiere profundizar el análisis a fin de lograr un modelo robusto y estable, en este sentido dejamos planteada la necesidad de explorar otras aristas, entre ellas, destacamos la evaluación del desempeño con distintas combinaciones de hiperparámetros y la experimentación con variaciones en el conjunto de datos. Entre los ajustes a los hiperparámetros encontramos la posibilidad de definir distintos tamaños del lote y tasa de aprendizaje, elegir distintos algoritmos de optimización y algoritmos de regularización y en el caso del conjunto de datos, se podría profundizar el estudio de las regiones de interés presentes en las imágenes que forman parte del conjunto de datos, mirando la heterogeneidad de características como un factor que puede afectar la convergencia y la capacidad de generalización del modelo [77].

4.7. Prueba del modelo

Este proceso es esencial para determinar cómo el modelo se desempeña al clasificar datos nuevos no vistos durante el entrenamiento, lo que permite inferir su comportamiento en escenarios del mundo real. Para evaluar el modelo se utilizan las métricas pérdida, exactitud, precisión y recall en el conjunto de datos prueba. El análisis de los resultados permitirá identificar si el modelo ha logrado un buen equilibrio entre la precisión y capacidad de generalización.

Para facilitar la prueba del modelo se utiliza el método *model.evaluate* del paquete Scikit.learn, pasando como parámetro el conjunto de datos del generador definido para la prueba (*test_generator*), los resultados se almacenan en la variable *results*.

```
results = model.evaluate(test_generator)
```

```
print(f"Resultados: {results}")
```

A continuación presentamos los resultados obtenidos para el dataset500 y dataset1300, dejando de lado los resultados del dataset40 por considerarlo de utilidad sólo para el análisis exploratorio inicial brindando el conocimiento necesario para definir la configuración inicial de hiperparámetros.

Un buen modelo está representado por valores bajos de pérdida, indicando que el modelo está realizando buenas predicciones y cometiendo pocos errores, junto a valores altos para la exactitud representando la proporción de predicciones correctas sobre el total de predicciones, indicando que el modelo está clasificando correctamente la mayoría de las muestras. La métrica precisión mide la proporción de predicciones positivas que son realmente positivas, donde un valor alto indica que cuando el modelo predice una clase positiva es muy probable que esa predicción sea correcta, y por último consideramos la sensibilidad o recall que mide la proporción de instancias positivas que fueron correctamente identificadas como positivas, donde un valor alto indica que el modelo es capaz de identificar la mayoría de los casos positivos. Destacamos que tanto la precisión como la sensibilidad son medidas muy importantes en el área de la ciberseguridad donde el costo de no detectar un caso positivo es muy alto (falso positivo).

Destacamos los resultados obtenidas para el conjunto de datos entrenado con el dataset1300: el valor de la pérdida fue 0,396, indicando una diferencia del 39,6% entre predicciones, la exactitud fue del 0,836%, es decir en el 83,6% predijo correctamente la clase, la precisión midió 0,857 indicando que de todas las imágenes que el modelo clasificó como phishing, el 85,7% eran efectivamente casos de phishing, o dicho de otra forma, cuando el modelo clasifica una imagen como phishing, tiene una probabilidad del 85,7% de estar en lo correcto y por último la sensibilidad midió 0,807 indicando que el modelo detectó el 80,7% de las imágenes etiquetadas como phishing en el conjunto de datos, es decir, de todas las imágenes que realmente eran phishing, el modelo identificó correctamente el 80,7%. Destacamos que los resultados obtenidos, vistos en el contexto de un proyecto de investigación, son razonables y alentadores para continuar el proyecto.

4.8. Predicciones del modelo

La etapa de predicción, tiene como objetivo probar el comportamiento del modelo en imágenes completamente nuevas, simulando el comportamiento en un escenario cercano a la realidad. Los datos provienen de un conjunto almacenado en Google Colab. El proceso inicia definiendo la ubicación de las imágenes dispuestas en Google Colab, luego se configura un generador incorporando la transformación de re-escalado, se crea la estructura para el etiquetado conteniendo las carpetas phishing y normal y se cargan las imágenes desde Google Colab, por último se realiza la predicción sobre las imágenes cargadas y se calculan las métricas (*precisión, exactitud y sensibilidad*) para analizar el desempeño del modelo. Este proceso permite verificar la capacidad de generalización del modelo y validar la confiabilidad en la detección de casos de phishing en situaciones prácticas.

```
# Realizar predicciones con el modelo
prediction_dir = 'data/predictions' # Carpeta de imágenes

os.makedirs(prediction_dir, exist_ok=True)

# Crear un generador para las predicciones

prediction_datagen = ImageDataGenerator(rescale=1./255)

# Crear la estructura de directorios

os.makedirs("data/predictions/phishing", exist_ok=True)

os.makedirs("data/predictions/normal", exist_ok=True)

from google.colab import files

# Subir imágenes de phishing

uploaded_phishing = files.upload()
```

```
for filename in uploaded_phishing.keys():
    os.rename(filename, f"data/predictions/phishing/{filename}")
# Subir imágenes normales
uploaded_normal = files.upload()
for filename in uploaded_normal.keys():
    os.rename(filename, f"data/predictions/normal/{filename}")
# Crear un generador para las imágenes de predicción
prediction_generator = prediction_datagen.flow_from_directory(
    prediction_dir,
    target_size=(224,224),
    batch_size=1,  # Procesar una imagen a la vez
    class_mode='binary',
    shuffle=False  # Para mantener el orden de las imágenes
)
# Realizar predicciones
predictions = model.predict(prediction_generator)
predicted_classes = (predictions > 0.5).astype(int)
# Obtener las etiquetas verdaderas a partir del generador
true_labels = prediction_generator.classes
# Calcular las métricas
```

```
accuracy = accuracy_score(true_labels, predicted_classes)

precision = precision_score(true_labels, predicted_classes)

recall = recall_score(true_labels, predicted_classes)

# Imprimir los resultados

print("Accuracy:", accuracy)

print("Precision:", precision)

print("Recall:", recall)

# Crear un DataFrame con los resultados

results_df =

pd.DataFrame(

{'Imagen': prediction_generator.filenames,

 'Etiqueta Verdadera': class_labels[true_labels],

 'Etiqueta Predicha': predicted_classes}

)

# Mostrar la tabla de resultados

print(results_df)
```

Para facilitar la evaluación del rendimiento utilizamos la matriz de confusión [78] donde se muestran las predicciones del modelo comparada con los valores reales. Los resultados obtenidos ofrecen una mirada sobre la capacidad del modelo para generalizar y su aplicabilidad en la detección de phishing.

```
# Calcular la matriz de confusión
```

```
cm = confusion_matrix(true_labels, predicted_classes)

# Configurar el gráfico

plt.figure(figsize=(8, 6))

sns.heatmap(cm,       annot=True,       fmt='d',       cmap='Blues',
xticklabels=class_labels, yticklabels=class_labels)



# Añadir etiquetas y título

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()
```

En la matriz se puede observar con claridad los valores obtenidos para los *verdaderos positivos* (VP), representando las instancias que el modelo predijo correctamente como positivas, en nuestro contexto los casos de phishing, los *verdaderos negativos* (VN), mostrando las instancias que el modelo predijo correctamente como negativas, en nuestro caso las imágenes correspondientes a los sitios normales, los falsos positivos (FP), aquellas instancias que el modelo clasificó erróneamente como positivas, es decir una falsa alarma y los falsos negativos (FN), cuanta las instancias que el modelo clasificó erróneamente como negativas, en el contexto de ciberseguridad, a este último elemento se le debe prestar especial atención, dado que representa los casos de phishing que el modelo no puedo detectar. En el análisis de la matriz para un modelo con un buen rendimiento se deberían encontrar valores altos en la diagonal principal, donde se muestran las predicciones correctas del modelo y valores bajos en los cuadros fuera de la diagonal, dado que estos cuadros muestran los errores en la clasificación.

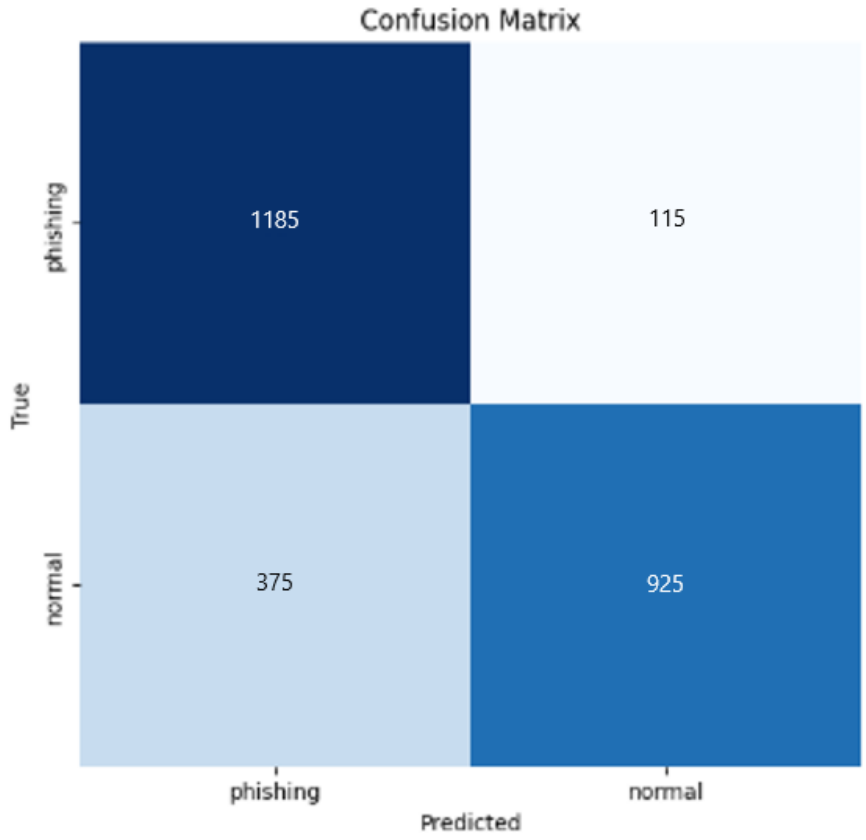

Figura 9. Matriz de confusión del dataset1300. Elaboración propia, en base a la librería seaborn.

En el gráfico se observan valores altos en la diagonal principal, indicando que el modelo presenta un buen desempeño, luego, la mirada sobre los valores fuera de la diagonal, permite identificar un llamado de atención, en el caso de los falsos negativos, cuyo número si bien es bajo en relación al resto de los valores de la matriz, es alto en el contexto de la ciberseguridad debido al costo asociado a la predicción como normal de un caso que es phishing.

4.9. Análisis de los resultados obtenidos

El modelo fue entrenado y evaluado utilizando tres conjuntos de datos de diferentes tamaños con el objetivo de analizar su capacidad de generalización, rendimiento y necesidad de recursos de cómputo a medida que aumenta la cantidad de datos. Las

métricas utilizadas son exactitud o accuracy, que mide la proporción de predicciones correctas sobre el total de ejemplos, la precisión o precision que representa la proporción de casos verdaderos positivos sobre los casos positivos predichos y la sensibilidad o recall, que representa la proporción de casos verdaderos positivos sobre el total de casos positivos reales.

El dataset40, con 40 imágenes en cada clase, 80 imágenes en total, se utilizó fundamentalmente para evaluar el consumo de recursos de computación y definir la infraestructura para entrenar y probar el modelo. En el caso del dataset500, con 1.000 imágenes, de las cuales 500 corresponden a sitios web de phishing y la misma cantidad de imágenes de sitios normales o sin phishing, obtuvo una exactitud de 82,22%, una precisión de 89,19% y la sensibilidad fué de 73,3%. El valor de precisión indica que el modelo presenta una baja tasa de falsos positivos, es decir, cuando predice phishing, en el 89,19% de las veces está en lo correcto. Sin embargo, un valor bajo de sensibilidad nos indica que el modelo tiene dificultades para detectar todos los casos de phishing, lo cual se traduce en un alto número de falsos negativos. El modelo muestra un desempeño aceptable, con capacidad para generalizar, pero aún presenta limitaciones en la detección de ataques de phishing.

El análisis de dataset1300, formado por 2.600 casos en total, 1.300 en cada clase, obtuvo una exactitud del 87,22%, una precisión del 92,40% y la sensibilidad midió 81,11%. El enriquecimiento del conjunto de datos ofrece mejores resultados en comparación con el dataset500. El aumento de la exactitud indica una mejora en la capacidad del modelo para realizar predicciones correctas; la mejora en la precisión y sensibilidad, indican mayor robustez del modelo en el contexto del problema en estudio vinculado a la ciberseguridad, donde la detección acertada junto a la minimización de los falsos negativos es un factor a tener en cuenta, observado en la brecha entre precisión y sensibilidad.

Dataset	Cantidad de imágenes	Exactitud	Precisión	Sensibilidad
Dataset500	1.000	0,8222	0,8919	0,7333

Dataset1300	2.600	0,8722	0,9240	0,8111

Tabla 1. Resumen de los resultados obtenidos en las métricas. Elaboración propia.

El análisis de las métricas revela una serie de tendencias, potencialmente relacionadas con el tamaño del dataset:

- **Exactitud**: al aumentar el tamaño del dataset, la exactitud mejora de 0.8222 a 0.8722, sugiriendo una mejor capacidad de generalización.
- **Precisión**: la métrica se incrementa con el tamaño del dataset, pasando de 0,8919 a 0.9240 en el dataset1300, indicando una mejora en la confiabilidad del modelo en la identificación de casos de phishing y un bajo nivel de falsos positivos.
- **Sensibilidad**: mejora con la ampliación del conjunto de datos, pasando de 0.7333 a 0.8111. Analizando el resultado en el conjunto de datos dataset1300, observamos que la sensibilidad presenta un valor menor que la precisión, sugiriendo que el modelo aún no detecta lo suficientemente bien todos los casos de phishing, incrementando la cantidad de falsos negativos, un aspecto crítico a analizar y mejorar.

A medida que aumenta el tamaño del dataset, el modelo mejora su capacidad de generalización, logrando resultados más confiables y métricas más estables. El modelo presenta resultados aceptables para la precisión, mostrando la efectividad para evitar falsas alarmas y garantizar predicciones confiables. Sin embargo, la detección completa de casos de phishing, medido en la sensibilidad, presenta oportunidades de mejora, especialmente considerando que la omisión de amenazas puede tener implicaciones críticas en el área de la ciberseguridad. Estas observaciones subrayan cómo el tamaño y la calidad del dataset influyen en el balance entre precisión y sensibilidad, así como en la capacidad del modelo para adaptarse y responder a desafíos más complejos.

Resultados de las predicciones

Anteriormente se presentaron los resultados del entrenamiento y prueba del modelo, adicionalmente se definió una etapa de predicción con el objetivo de validar el desempeño del modelo sobre datos no vistos, ni en el entrenamiento, ni en la prueba, simulando las

condiciones de un entorno real. Se trabajó con un generador, para facilitar la carga y transformación de los datos y la predicción de las imágenes. Los resultados de las predicciones fueron alentadoras, permitiendo validar el desempeño obtenido en el entrenamiento y prueba. Se realizaron 90 predicciones mostrando un buen rendimiento, alcanzando una precisión de 89,5%, indicando que el modelo presenta un buen desempeño en el contexto de un proyecto de investigación.

Limitaciones y Problemas Detectados

Pensando en la continuidad del trabajo de investigación abordado o en futuras líneas relacionadas con el aprendizaje automático y el procesamiento de imágenes, consideramos oportuno comentar las limitaciones y escollos que encontramos.

- **Dificultad para extraer patrones en imágenes similares:** el modelo mostró dificultades al analizar imágenes de phishing visualmente similares a sitios legítimos, dificultando la detección de las regiones de interés, resultando en un elevado número de falsos negativos. El problema indica la necesidad de prestar atención a los resultados de las métricas en relación con el problema en estudio, por ejemplo en nuestro caso nos movemos en el contexto de la ciberseguridad donde debemos prestar atención a la precisión y sensibilidad, con el objetivo de minimizar los falsos negativos.
- **Desbalance del conjunto de datos**: el análisis del dataset original mostraba un leve desbalance entre las clases (imágenes de sitios web normales y sitios web de phishing), realizando ajustes con el objetivo de abordar los potenciales problemas durante el entrenamiento y prueba del modelo, como el sobreajuste, sesgo en la clasificación hacia la clase mayoritaria en detrimento de la precisión en las clases minoritarias. La búsqueda del balance en las clases se enfoca hacia una representación más equitativa en consecuencia la búsqueda de un mejor desempeño del modelo.
- **Limitaciones de los recursos disponibles**: el abordaje de los problemas vinculados con el aprendizaje automático requieren poder de cómputo, la consecuencia de recursos limitados es el incremento en el tiempo de procesamiento, pasando de horas a semanas, de acuerdo al tamaño del conjunto de datos y la arquitectura del

modelo. En nuestro caso, intentamos trabajar en una notebook sin GPU, pero debimos dejarla de lado debido al tiempo requerido para la carga de datos y entrenamiento del modelo. Como alternativa, consideramos el uso de Google Colab, donde disponemos de GPU para la ejecución. El entorno de programación presenta restricciones operativas, como el uso de lotes pequeños y un número reducido de épocas durante el entrenamiento. Estas limitaciones alertan sobre la necesidad de optimizar los recursos disponibles analizando las facilidades que brindan las librerías de alto nivel como keras, y, en caso que el problema requiera aumentar el tamaño del conjunto de datos y/o arquitecturas complejas se podrá avanzar gradualmente, planteando objetivos abordables con los recursos disponibles obteniendo resultados válidos para un contexto de investigación.

5. Conclusiones

Durante el desarrollo del proyecto se implementaron con éxito técnicas de aprendizaje profundo orientadas a identificar patrones en imágenes compatibles con el ataque informático phishing en sitios web, una de las amenazas más críticas en el ámbito de la ciberseguridad. El enfoque adoptado permitió analizar de manera eficiente imágenes complejas y validar la capacidad de las redes neuronales convolucionales para aportar herramientas avanzadas en la detección de ataques.

Entre los resultados más relevantes, destacamos los beneficios del modelo VGG16, permitiendo acortar el esfuerzo computacional requerido para obtener un modelo compatible con el problema. La utilización del modelo pre-entrenado y la técnica de la transferencia de aprendizaje demostró ser una estrategia eficaz, logrando un modelo apto para la clasificación de imágenes de phishing con un desempeño acorde al objetivo del trabajo.

Asimismo, el estudio permitió aprender técnicas para abordar problemas clásicos relacionados con el aprendizaje automático, como el procesamiento y la transformación del conjunto de datos, y el análisis de las métricas obtenidas para encarar problemas como el sesgo, sobreajuste y los falsos positivos, entre ellos destaca la importancia de la calidad y organización del conjunto de datos en la capacidad de generalización del modelo. Los ajustes realizados sobre las imágenes mostraron un mejor desempeño y confiabilidad del modelo logrando reducir la tasa de falsos positivos.

En cuanto a la detección de imágenes complejas, el modelo mostró resultados prometedores, sobre todo en la identificación de casos donde las regiones de interés compatibles con phishing estaban difusas. Sin embargo, estos casos complejos presentan desafíos, impactando en la tasa de falsos negativos, resaltando la importancia de continuar en la línea de investigación analizando las oportunidades de llevar adelante estudios que busquen optimizar la capacidad de generalización del modelo.

Por otro lado, se identificaron limitaciones técnicas durante el desarrollo, principalmente debido a la imposibilidad de contar con GPU en las estaciones de trabajo, a lo cual se sumaron las restricciones para el acceso en la modalidad sin costo impuestas en las plataformas como Google Colab. Estas restricciones limitan el acceso a arquitecturas complejas, o también limitan el tamaño de los conjuntos de datos, junto con el tiempo disponible para el uso. Las restricciones anteriores limitaron la exploración de las causas y posibles soluciones vinculadas a lograr un modelo más robusto y con un mejor desempeño.

En conclusión, el presente trabajo demostró la posibilidad de aportar herramientas que colaboren con la lucha contra las amenazas informáticas a partir de técnicas de aprendizaje profundo, en particular se exploró el uso de un clasificador binario basado en la arquitectura VGG16 entrenado para la detección de patrones de phishing de sitios web en imágenes. Siguiendo una metodología basada en las actividades de preprocesamiento, entrenamiento, evaluación, ajuste y predicción se lograron resultados prometedores en el contexto de la investigación. No obstante, persisten desafíos que abren nuevas oportunidades, entre ellos podemos mencionar la necesidad de ampliar la diversidad y representatividad del conjunto de datos, el análisis de las alternativas para mejorar la detección de imágenes ambiguas y la exploración de arquitecturas híbridas para optimizar el rendimiento del modelo.

5.1. Próximos pasos

Consideramos oportuno destacar las líneas de trabajo que podrían ser exploradas con el objetivo de dar continuidad al trabajo de investigación desarrollado:

Mejorar la calidad del conjunto de datos: incorporar transformaciones que busquen representar las características de las imágenes capturadas con distintos dispositivos, por ejemplo, se puede explorar el resultado de las técnicas de aumento de datos que incluyan desenfoque (blur), ajustes de brillo y contraste. Estas transformaciones apuntan a aumentar la diversidad y representatividad de los patrones presentes en el conjunto de datos, aumentando la capacidad del modelo para generalizar en escenarios complejos;

también se pueden sumar imágenes provenientes de distintas fuentes, como sitios de phishing recientemente generados y variaciones regionales en el diseño de interfaces. Adicionalmente, se podría considerar el aporte de técnicas de generación sintética, como las GAN (Generative Adversarial Networks) [79], para enriquecer el conjunto de datos con ejemplos más complejos y desafiantes, fortaleciendo así la capacidad del modelo para enfrentar las nuevas y variadas técnicas de phishing web.

Exploración de técnicas de aprendizaje profundo: probar métodos automáticos para la selección de subconjuntos representativos de datos, optimizando el proceso de entrenamiento y validación. Analizar el resultado de técnicas para la evaluación automática de hiperparámetros, como GridSearchCV, destinadas a buscar las configuraciones óptimas para el modelo sin depender de ajustes manuales, contribuyendo al logro de un proceso más optimizado enfocado en la construcción del modelo.

Exploración de la aplicabilidad en contextos reales: implementar el modelo en un entorno de prueba de concepto que emule condiciones reales de ciberseguridad, evaluando su capacidad para detectar actividades de phishing en tiempo real.

6. Bibliografía

1. F. Dalgıç, A. Bozkir, and M. Aydos, "Phish-IRIS: A New Approach for Vision Based Brand Prediction of Phishing Web Pages via Compact Visual Descriptors," in *2018 IEEE International Symposium on Multimedia Information & Security Technologies (ISMSIT)*, pp. 1-6, doi: 10.1109/ISMSIT.2018.8567299. 2018.
2. Kaspersky, "Nueva epidemia: el phishing se sextuplicó en América Latina con el reinicio de la actividad económica y el apoyo de la IA," Kaspersky Lab, 2024, [en línea] Disponible: https://latam.kaspersky.com/blog/panorama-amenazas-latam-2023/26586/
3. Verizon, "Data Breach Investigations Report (DBIR)", 2023, [online] Available: https://www.verizon.com/business/resources/reports/dbir/
4. N. Gutierrez, “Estadísticas de Phishing en Argentina”. *PreyProject Blog*. 2023, Disponible: https://preyproject.com/es/blog/phishing-en-latinoamerica
5. Y. LeCun, Y. Bengio and G. Hinton, “Deep learning”, Nature, vol. 521, no. 7553, pp. 436–444, May. 2015.
6. M. Adebowale, K. Lwin y M. Hossain, “Esquema de detección de phishing inteligente utilizando algoritmos de aprendizaje profundo”, 2023.
7. F. Alsubaie, A. Almazroi y N. Ayub, “Mejora de la detección de phishing: un nuevo marco híbrido de aprendizaje profundo para la investigación forense de delitos cibernéticos”, 2023.
8. G. Benitez y D. Kanaan, “Inteligencia artificial avanzada”, Spanish Edition. Barcelona, España: Editorial UOC, 2013.
9. I. Goodfellow, Y. Bengio & A. Courville, “Deep Learning”, MIT Press, 2016.
10. R. Duda, P. Hart & D. Stork, “Pattern Classification”, Wiley, 2001.
11. N. Do, A. Selamat, O. Krejcar, et al., "Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions," in IEEE Access, vol. 10, pp. 36429-36463, 2022, doi: 10.1109/ACCESS.2022.3151903.
12. L. Fausett, “Fundamentals of neural networks: architectures, algorithms, and applications”. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.
13. Artificial Intelligence Index: Anual Report, 2018, [online]. Available: http://bit.ly/34cPs6J, [Accessed Nov. 2024].

14. M. Koch, "The neurobiology of startle" Prog Neurobiol, 1999, doi: 10.1016/s0301-0082(98)00098-7.
15. V. Passricha, R. Aggarwal, “A Hybrid of Deep CNN and Bidirectional LSTM for Automatic Speech Recognition”, Journal of Intelligent Systems, mar. 2019, doi: 10.1515/jisys-2018-0372
16. D. Quirumbay Yagual, C. Castillo Yagual y I. Coronel Suarez, “Una revisión del aprendizaje profundo aplicado a la ciberseguridad”. RCTU, vol.9, n.1, pp.57-65. ISSN 1390-7697, 2022, doi: 10.26423/rctu.v9i1.671.
17. UCF CRCV, “Center for Research in Computer Vision,” 2019 [online]. Available: http://bit.ly/2qEA9F5. [Accesed: Nov. 21, 2024]
18. S. Ahmad, "Across the Spectrum In-Depth Review AI-Based Models for Phishing Detection", in IEEE Open Journal of the Communications Society, doi: 10.1109/OJCOMS.2024.3462503.
19. M. Batta, “Machine Learning Algorithms - A Review”, *International Journal of Science and Research (IJSR)*, 2020, doi: 10.21275/art20203995.
20. M. Hall, F. Doshi-Velez and J. Kevitt, “Automated machine learning: Review of the state of the art and opportunities for healthcare”, *Artificial Intelligence in Medicine*, vol. 101, no. 1, pp. 101822, feb. 2020. doi: 10.1016/j.artmed.2020.101822.
21. V. Passricha, R. Aggarwal, “A Hybrid of Deep CNN and Bidirectional LSTM for Automatic Speech Recognition, Journal of Intelligent Systems”, mar. 2019. doi: 10.1515/jisys-2018-0372.
22. N. Jaouedi, N. Boujnah and M. Salim Bouhlel, “A new hybrid deep learning model for human action recognition, Journal of King Saud University”, Computer and Information Sciences, 2019.
23. K. Maninis, “Convolutional Oriented Boundaries: From Image Segmentation to HighLevel Tasks”, TPAMI, 819-833, apr. 2018.
24. MIT Task Force on the work of the future, “*The Work of the Future: Shaping Technology and Institutions”*. Fall 2019 Report. 2019.
25. L. Fausett, “Fundamentals of neural networks: architectures, algorithms, and applications”. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.

26. International Association for Pattern Recognition. *Pattern Recognition Letters*. Elsevier. 2019.
27. E. Aldakheel, et al., “A Deep Learning-Based Innovative Technique for Phishing Detection in Modern Security with Uniform Resource Locators.” Sensors (Basel, Switzerland), 2023.
28. A. Karpathy, et al., "Large-scale Video Classification with Convolutional Neural Networks", [online]. Available: https://stanford.io/2QIK4nW, [Accessed: nov. 21, 2019].
29. N. Jaouedi, N. Boujnah, M. Salim Bouhlel, “A new hybrid deep learning model for human action recognition”, Journal of King Saud University, Computer and Information Sciences, 2019.
30. T. Ige, “An investigation into the performances of current state-of-the-art models for phishing detection, 2024, [online]. Available: https://arxiv.org/abs/2411.16751.
31. Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, et al., “A deep learning-based phishing detection system using CNN, LSTM, and LSTM-CNN,” *Electronics*, vol. 12, no. 1, p. 232, jan. 2023, doi: 10.3390/electronics12010232.
32. P. González, “*Ethical Hacking: Teoría y práctica para la realización de un pentesting”*, 3ª ed., pp. 246. Madrid, España: 0xWord. ISBN: 978-84-09-48554-3. 2023.
33. I. Sommerville, I. “*Ingeniería de software”,* 9na ed, Pearson. 2011.
34. Ö Aslan, S. Aktug, M. Semih et al., “A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions”. Electronics. 12. 1-42, 2023, doi: 10.3390/electronics12061333.
35. V. Batagelj, H. Bock and A. Ferligoj, “Data Science and Classification”, Springer, 2006.
36. R. Ayeni, A. Adebiyi, J. Okesola, et al., “Phishing Attacks and Detection Techniques: A Systematic Review”. 1-17, 2024, doi: 10.1109/SEB4SDG60871.2024.10630203.
37. Dirección Nacional de Ciberseguridad, “*Phishing: Una guía y un glosario para conocer sus modalidades y prevenirlas”*. Jefatura de Gabinete de Ministros, 20210, [en línea], Disponible en https://www.argentina.gob.ar/jefatura/innovacion-publica/ssetic/direccion-nacional-ciberseguridad/informes-de-la-direccion-1.

38. K. Thakur, M. Ali, M. Obaidat and A. Kamruzzaman, “A Systematic Review on Deep-Learning-Based Phishing Email Detection”. Electronics, 12, 4545, 2023, doi: 10.3390/electronics12214545.
39. V. Yazhmozhi, “Anti-phishing System using LSTM and CNN”, *2020 IEEE International Conference on Innovative Technologies and Applications (INOCON),* pp. 1–6, IEEE, 2020, doi: 10.1109/inocon50539.2020.9298298.
40. S. Dambe, S. Gochhait and S. Ray, “The Role of Artificial Intelligence in Enhancing Cybersecurity and Internal Audit”, In *2023 IEEE Asian Conference on Engineering and Computer Science (AECE), 2023,* doi: 10.1109/aece59614.2023.10428353.
41. Z. Zhang, “Explainable artificial intelligence applications in cyber security: State-of-the-art”, *IEEE Access*, 2022, doi: 10.1109/access.2022.3204051.
42. F. Silva, T. Pereira, J. Morgado, J. Frade, et al., “EGFR Assessment in Lung Cancer CT Images: Analysis of Local and Holistic Regions of Interest Using Deep Unsupervised Transfer Learning", *IEEE Access*, 2021.
43. K. Thakur, M. Ali, M. Obaidat, A. Kamruzzaman, “A Systematic Review on Deep-Learning-Based Phishing Email Detection”. Electronics, 12, 4545, 2023, doi: 10.3390/electronics12214545.
44. OpenAlex, "OpenAlex: The open catalog to the global research system." [online]. Available: https://openalex.org/
45. M. Gao, H. Hou, C. Wang, “Machine Learning and Deep Learning Methods for Cybersecurity”, *IEEE Access*, 6, 35365-35381, 2018, doi:10.1109/ACCESS.2018.2836950.
46. Sarker, A. Kayes, S. Badsha, H. Alqahtani, et al., “Cybersecurity data science: an overview from machine learning perspective”, *Journal of Big Data*, 7(1), 2020, doi:10.1186/s40537-020-00318-5.
47. K. Shaukat, S. Luo, V. Varadharajan, et al., "Performance Comparison and Current Challenges of Using Machine Learning Techniques in Cybersecurity", Energies, vol. 13, no. 10, pp. 2509, 2020, doi: 10.3390/en13102509.
48. C. Zonyfar, J. Lee and J. Kim, “HCNN-LSTM: Hybrid Convolutional Neural Network with Long Short-Term Memory Integrated for Legitimate Web Prediction”, *Journal of Web Engineering,* 2023.

49. S. Bamber, A. Katkuri, S. Sharma, et al., “A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system”, Computers & Security, 148, 2024, doi: 10.1016/j.cose.2024.104146.
50. A. Nazir, J. He, N. Zhu, et al., “A deep learning-based novel hybrid CNN-LSTM architecture for efficient detection of threats in the IoT ecosystem”, In Shams Engineering Journal, 15(7), 102777, 2024, doi: 10.1016/j.asej.2024.102777.
51. J. Smith, J. Doe, “The Comparison of Cybersecurity Datasets”, Data, 1-10, 2022, doi: 10.3390/data7020022.
52. M. Ozkan Okay, E. Akin, Ö. Aslan, et.al., “A Comprehensive Survey: Evaluating the Efficiency of Artificial Intelligence and Machine Learning Techniques on Cyber Security Solutions”, *IEEE Access*, 1-10, 2024, doi: 10.1109/ACCESS.2024.3333333.
53. I. Sarker, M. Furhad and R. Nowrozy, “AI-Driven Cybersecurity: An Overview Security Intelligence Modeling and Research Directions”. Computer Science, [online]. vol. 2, num. 3, págs. 1-18. ISSN 2662-995X, 2023, doi: 10.1007/s42979-021-00557-0.
54. M. Amanullah, V. Selvakumar, A. Jyot, et al., “CNN based prediction analysis for web phishing prevention”, In 2022 International Conference on Edge Computing and Applications (ICECAA), 2022.
55. R. Zaimi, M. Hafidi and M. Lamia, “A deep learning approach to detect phishing websites using CNN for privacy protection”, *Intelligent Decision Technologies,* 2023.
56. F. Trad and A. Chehab, “Prompt Engineering or Fine-Tuning? A Case Study on Phishing Detection with Large Language Models”. *Machine Learning and Knowledge Extraction,* 2024.
57. L. Passos, D. Jodas, K. Costa, et al., “A review of deep learning-based approaches for deepfake content detection”, *Expert Systems,* 2024.
58. P. Loh, A. Lee and V. Balachandran, “Towards a Hybrid Security Framework for Phishing Awareness Education and Defense”, *Future Internet,* 2024.
59. Google Colaboratory. Google Research. (n.d.). A free, cloud-based Jupyter notebook environment for Python code execution, [online] Available: https://colab.research.google.com/
60. Python Software Foundation. Python programming language (n.d.). A high-level, general-purpose programming language, [online] Available: https://www.python.org/

61. TensorFlow. TensorFlow (n.d.). An open-source library for numerical computation using data flow graphs, [online] Available: https://www.tensorflow.org/
62. Keras. Keras (n.d.). A high-level neural network API built on top of TensorFlow, [online] Available: https://keras.io/
63. NumPy. NumPy (n.d.). A fundamental library for scientific computing in Python. [online] Available: https://numpy.org/
64. Pandas. Pandas (n.d.). A high-performance data analysis and manipulation library for Python, [online] Available: https://pandas.pydata.org/
65. Matplotlib. Matplotlib: A 2D graphics environment (n.d.), [online] Available: https://matplotlib.org/stable/
66. Scikit-learn. Scikit-learn: Machine Learning in Python (n.d.). A comprehensive toolbox for machine learning tasks in Python, [online] Available: https://scikit-learn.org/stable/
67. K. Simonyan, A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, Proceedings of the International Conference on Learning Representations, 2014.
68. J. Deng, W. Dong, R. Socher, et. al., ”ImageNet: A Large-Scale Hierarchical Image Database”, in CVPR 2009, IEEE, 2009.
69. E. Rezende, G. Ruppert, T. Carvalho, et al., "Malicious Software Classification Using VGG16 Deep Neural Network's Bottleneck Features", Advances in Intelligent Systems and Computing, 2018.
70. F. Dalgıç, A. Bozkir and M. Aydos, “Phish-IRIS: A New Approach for Vision Based Brand Prediction of Phishing Web Pages via Compact Visual Descriptors”, 2018, doi: 10.1109/ISMSIT.2018.8567299.
71. A. Krizhevsky, I. Sutskever and G. Hinton, “Imagenet classification with deep convolutional neural networks”, Advances in neural information processing systems, 25, 1097-1105, 2012.
72. D. Kingma and J. Ba, “Adam: A method for stochastic optimization”, arXiv preprint arXiv:1412.6980, 2014.
73. I. Goodfellow, Y. Bengio and A. Courville, “Deep learning”, MIT press, 2016.
74. M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks”, Information processing & management, 45(4), 427-437, 2009.

75. A. Geron, “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow”, O'Reilly Media, 2019.
76. M. Jason, “Mastering Hyperparameter Tuning with GridSearchCV”, *Towards Data Science, 2023,* [online] Available https://towardsdatascience.com/mastering-hyperparameter-tuning-with-gridsearchcv-9f8692859d2e.
77. J. Hafeez, “Evaluating Feature Extraction Methods with Synthetic Noise Patterns for Image-Based Modelling of Texture-Less Objects”, Remote Sensing, vol. 12, no. 23, p. 3886, 2020, doi: 10.3390/rs12233886.
78. T. Hastie, R. Tibshirani and J. Friedman, “The elements of statistical learning: data mining, inference, and prediction”, 2nd ed ,Springer, 2009.
79. I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., “Generative adversarial nets”, In *Advances in neural information processing systems*, pp. 2672-2680, 2014.