

UNIVERSIDAD CENTRAL “MARTA ABREU” DE LAS VILLAS  
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN



DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TRABAJO DE DIPLOMA

**Propuesta de un nuevo método de explicabilidad  
para interpretar los resultados de modelos  
entrenados para la clasificación de imágenes**

*Autor:*

*Esther María Martín Hernández*

*Tutores:*

*Dra. C. María Matilde García Lorenzo*

*Dra. C. Marilyn Bello García*

*Consultor:*

*MSc. Alejandro Ramón Hernández*

2023

UNIVERSIDAD CENTRAL “MARTA ABREU” DE LAS VILLAS  
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN



DEPARTMENT OF COMPUTER SCIENCE

DIPLOMA THESIS

**Proposal of a new explainability method to  
interpret the results of trained models for image  
classification**

*Author:*

*Esther María Martín Hernández*

*Tutors:*

*Dr.Sc. María Matilde García Lorenzo*

*Dr.Sc. Marilyn Bello García*

*Consultant:*

*MSc. Alejandro Ramón Hernández*

2023



## ACTA DE CONFORMIDAD PARA ESTUDIANTES DE PREGRADO

Universidad Central "Marta Abreu" de Las Villas

Por una parte: Esther María Martín Hernández, estudiante de la carrera de: Lic. Ciencia de la Computación en la facultad de: Matemática Física y Computación, en lo adelante **EL ESTUDIANTE**. Con número de identidad permanente: 01092372735 o pasaporte: \_\_\_\_\_. Y por otra parte Gerardo Hernández Jefe del Departamento Docente de: Computación

en la ya mencionada facultad, en lo adelante **EL JEFE DE DEPARTAMENTO**, y María Matilde García Lorenzo y Marilyn Bello García profesor(es) encargado(s) de tutorar el Trabajo de Diploma **DEL ESTUDIANTE**, en lo adelante **EL TUTOR**.

Reconocen que:

- I. A **EL ESTUDIANTE** se le ha aprobado como tema de investigación para su Trabajo de Diploma el titulado Propuesta de un nuevo método de explicabilidad para interpretar los resultados de modelos entrenados para la clasificación de imágenes
- II. **EL ESTUDIANTE** no divulgará información concerniente a la investigación, tanto durante el desarrollo como tras la culminación de esta sin la debida autorización **DEL TUTOR** o **EL JEFE DE DEPARTAMENTO**.
- III. Que el Trabajo de Diploma fruto de la labor investigativa de **EL ESTUDIANTE** y la asesoría de **EL TUTOR**, resulta de **TITULARIDAD EXCLUSIVA** de la Universidad Central "Marta Abreu" de las Villas.
- IV. **EL ESTUDIANTE** una vez aprobada su tesis para la defensa, depositará una copia electrónica de la misma en el Repositorio Digital Institucional de la Universidad Central "Marta Abreu" de Las Villas.
- V. A partir de la defensa y aprobación del Trabajo de Diploma, la publicación total, parcial o la elaboración de cualquier obra que se derive de esta investigación por parte de **EL ESTUDIANTE**, contará con la coautoría de **EL TUTOR** y viceversa, resultando de referencia obligada esta obra en cualquier otra que se elabore. El incumplimiento de esta cláusula, puede llevar consigo el inicio de procesos de plagio. Todo lo anterior de acuerdo a la normativa de Derecho de Autor vigente en Cuba.

Y para que así conste se firma la presente en la Universidad Central "Marta Abreu" de Las Villas, a los 7 días del mes de diciembre del año 2023.

EL ESTUDIANTE

JEFE DE DEPARTAMENTO

TUTOR

TUTOR

*"No importa cuán difícil o imposible parezca, no apartes la vista de tu objetivo"*

—Eiichiro Oda

**DEDICATORIA**

*A mis padres, por el apoyo incondicional y constante a lo largo de mi vida.*

*A mi hermano, porque cerca o lejos es mi brújula y motivación.*

*A Daniel, por llenarme de su luz cada día con su paciencia, dedicación y aliento.*

**AGRADECIMIENTOS**

*A los pilares de mi vida: mis padres, mi hermano y mi esposo.  
A mis tías, mis tíos, mi hermana, mis sobrinas lindas y a toda mi familia por estar  
siempre, preocuparse y apoyarme en todo.  
A Bety por concederme el placer de sentir que es como una hermana para mí. También a  
sus padres y su familia por acogerme y hacerme sentir parte de ella.  
A mis compañeros que son casi familia después de estos últimos años viviendo  
experiencias juntos.  
A mi pandilla cienfueguera por estar para mí, darme aliento y confianza en cada  
"reunión".  
Especialmente a mis tutores María Matilde y Marilyn por su guía y consejos precisos en  
todo momento  
Y a todas las personas que de una forma u otra han contribuido en mi progreso hasta  
aquí:  
¡¡Muchas gracias!!*

## **Resumen**

En la actualidad, las redes neuronales han revolucionado diversos sectores gracias a su capacidad para comprender relaciones complejas en grandes conjuntos de datos. Sin embargo, la opacidad en la toma de decisiones de estos modelos ha generado desconfianza, impulsando la necesidad de la Inteligencia Artificial Explicable. La explicabilidad, en este contexto, se presenta como un componente esencial para superar los obstáculos mencionados, pero su implementación sigue siendo un desafío significativo. La falta de métodos concretos para evaluar la calidad de las explicaciones ha intensificado la necesidad de enfoques robustos y confiables capaces de proporcionar claridad y veracidad en las interpretaciones. En respuesta a este desafío se propone el diseño de un nuevo enfoque para interpretar las decisiones de modelos de clasificación de imágenes como VGG19, basado en redes neuronales y utilizando entre la definición de sus capas una función de perturbación. Se detalla la implementación de la propuesta, así como de las funciones auxiliares. El enfoque propuesto se evalúa en diferentes casos de estudio a partir de realizar configuraciones específicas al conjunto Food101, explorando de esa forma el método en diferentes escenarios se observó cierta consistencia en la interpretación de los mapas de relevancia. La comparación entre los casos de estudio reveló cambios específicos a realizar para mejorar la calidad de las explicaciones proporcionadas. Se obtuvieron resultados adecuados para una etapa experimental y se reconoce la necesidad de futuras investigaciones para refinar los escenarios de prueba y profundizar en la comprensión de las interacciones entre los parámetros del modelo, la función de perturbación y la interpretación de la explicación.

## **Abstract**

Nowadays, neural networks have revolutionized various sectors thanks to their ability to understand complex relationships in large data sets. However, the opacity in the decision-making of these models has generated distrust, driving the need for Explainable Artificial Intelligence. Explainability, in this context, is presented as an essential component to overcome the aforementioned obstacles, but its implementation remains a significant challenge. The lack of concrete methods to assess the quality of explanations has intensified the need for robust and reliable approaches capable of providing clarity and veracity in interpretations. In response to this challenge, we propose the design of a new approach to interpret the decisions of image classification models such as VGG19, based on neural networks and using a perturbation function between the definition of its layers. The implementation of the proposal is detailed, as well as the auxiliary functions. The proposed approach is evaluated in different case studies based on making specific configurations to the Food101 set, thus exploring the method in different scenarios some consistency was observed in the interpretation of the relevance maps. The comparison between the case studies revealed specific changes to be made to improve the quality of the explanations provided. Adequate results were obtained for an experimental stage and it is recognized the need for future research to refine the test scenarios and deepen the understanding of the interactions between the model parameters, the perturbation function and the interpretation of the explanation.

# Índice general

<b>Resumen</b>	<b>I</b>
<b>Abstract</b>	<b>II</b>
<b>Introducción</b>	<b>1</b>
<b>1. Aspectos Teóricos</b>	<b>6</b>
1.1. Inteligencia Artificial . . . . .	6
1.2. Aprendizaje Automático . . . . .	7
1.3. Redes Neuronales . . . . .	10
1.3.1. Fases de modelación con redes neuronales . . . . .	11
1.3.2. Topología de redes neuronales . . . . .	12
1.4. Aprendizaje Profundo . . . . .	14
1.4.1. Redes Neuronales Profundas . . . . .	15
1.4.2. Redes Neuronales Profundas Convolucionales . . . . .	17
1.4.3. Modelos de redes convolucionales . . . . .	22
1.5. Inteligencia Artificial Explicable . . . . .	23
1.5.1. Explicabilidad . . . . .	25
1.5.2. Herramientas actuales para la explicabilidad . . . . .	28
1.6. Conclusiones parciales . . . . .	31
<b>2. Implementación del nuevo método</b>	<b>33</b>
2.1. Descripción general del modelo . . . . .	33
2.1.1. Arquitectura de la red y particularidades a tener en cuenta en cada capa . . . . .	35
2.2. Implementación . . . . .	36
2.2.1. Análisis de la complejidad computacional . . . . .	39
2.3. Alcance y limitaciones del método propuesto . . . . .	40
2.4. Diferencias con respecto a los enfoques actuales . . . . .	41
2.5. Conclusiones parciales . . . . .	41
<b>3. Evaluación del método propuesto en los casos de estudio</b>	<b>43</b>
3.1. Descripción del conjunto de datos de entrada . . . . .	43
3.2. Transformaciones . . . . .	44
3.3. Caso de estudio 1 . . . . .	45

3.4. Caso de estudio 2 . . . . .	47
3.5. Caso de estudio 3 . . . . .	49
3.6. Comparación de los casos de estudio . . . . .	51
3.7. Conclusiones parciales . . . . .	52
<b>Conclusiones</b>	<b>54</b>
<b>Recomendaciones</b>	<b>55</b>
<b>Referencias</b>	<b>56</b>

# Introducción

Con la gradual evolución del tiempo, los problemas y las expectativas de la vida humana se han vuelto cada vez más complejos, lo que trae consigo la necesidad de un arduo trabajo para encontrar soluciones correspondientes. Este enfoque de actitud de resolución de problemas finalmente ha llevado a la humanidad a la era tecnológica actual, donde se realizan constantes intentos por hacer que las máquinas alcancen una inteligencia similar a la humana.

Desde sus primeros avances hasta la actualidad, la Inteligencia Artificial (IA o AI, del inglés Artificial Intelligence) ha encontrado múltiples aplicaciones en diversas áreas, relacionándose esta a las imitaciones por máquinas de funciones cognitivas asociadas a las mentes humanas como aprender y resolver problemas. Es por eso que se han logrado importantes capacidades en este campo como comprender exitosamente el habla humana, competir en alto nivel en sistemas de juegos estratégicos, integrar autos autónomos y sistemas de diagnóstico médico, realizar enrutamiento inteligente, simulaciones militares, interpretar datos complejos, detectar emociones en el reconocimiento de expresiones faciales y el procesamiento del lenguaje natural para comprender y empatizar con las emociones humanas, clasificar y etiquetar automáticamente las imágenes, permitiendo la organización y búsqueda eficiente de grandes conjuntos de datos visuales, entre muchas otras (Ongsulee, 2017).

Estas diversas capacidades, se han logrado a través del avance en los numerosos subcampos de la IA, entre los que ha tenido un gran impacto el Aprendizaje Automático o Automatizado (ML, del inglés Machine Learning) y el Aprendizaje Profundo (DL, del inglés Deep Learning), los cuales se enfocan en desarrollar algoritmos complejos y modelos que permiten a las máquinas aprender y tomar decisiones basadas en los datos (Ongsulee, 2017).

En este sentido, los algoritmos de ML analizan los datos, aprenden de estos y luego aplican lo que han aprendido para hacer predicciones o tomar decisiones informadas, logrando con el entrenamiento a medida que se exponen a más datos a lo largo del tiempo, mejorar su rendimiento sin programación explícita. Estos algoritmos se dividen en grupos, pueden ser supervisados, no supervisados, semi-supervisados o reforzados, según el tipo de datos y el enfoque de este (Taye, 2023). Además, existen varios enfoques de ML y es importante tener en cuenta que no son mutuamente excluyentes porque se pueden combinar para crear modelos más complejos. Entre ellos se encuentran el aprendizaje perezoso, el aprendizaje basado en árboles de decisión, el aprendizaje basado en reglas y el aprendizaje basado en redes neuronales o también conocido como aprendizaje conexionista. Este último es un tipo de método en el que el sistema aprende de un conjunto de nodos interconectados que

simulan la función de las neuronas en el cerebro (Dietterich et al., 2008).

Del mismo modo el aprendizaje profundo constituye un subconjunto del aprendizaje automático conexionista y los algoritmos de este aprendizaje requieren grandes cantidades de datos, incluidos datos diversos y no estructurados, para aprender y mejorar sus resultados. Esto hace que tenga dos ventajas principales: poder manejar volúmenes masivos de datos y poder mejorar los resultados a través de la repetición, sin intervención humana. (Grieve, 2023).

En general, según (Carvalho et al., 2019), los modelos de aprendizaje automático y aprendizaje profundo a menudo se consideran “cajas negras”, ya que su lógica interna y sus procesos de toma de decisiones no son fácilmente interpretables por los humanos y esta falta de interpretabilidad puede ser problemática, en especial en dominios sensibles como la atención médica, donde es importante comprender cómo un modelo llegó a su decisión.

Dado estos desafíos, la construcción de métodos robustos y prácticos para explicar las decisiones de un determinado modelo, se ha desarrollado en un área de investigación propia, la IA explicable (XAI, del inglés Explainable Artificial Intelligence) la cual busca mejorar el proceso de entrenamiento, las representaciones aprendidas y las decisiones con explicaciones interpretables por humanos (Samek et al., 2021).

Para (Lusim and Marchesano, 2023) dentro de la explicabilidad de modelos, existen los modelos interpretables como árboles de decisión, modelos basados en reglas y modelos de regresión lineal, entre otros, cuyos procesos de toma de decisión son transparentes, es decir es posible comprender la secuencia de pasos ejecutados para llegar a su respuesta. Y por otro lado están los modelos no interpretables, ya mencionados como cajas negras. En el caso de los modelos cajas negras existen diferentes enfoques para explicar sus decisiones. La literatura los divide en, explicaciones globales o locales, con el fin de determinar qué variables contribuyen al modelo para tomar decisiones y determinar cómo afectan los valores de esas variables a las predicciones, permitiendo abordar la falta de profundidad y transparencia. Esta puede también ayudar a los desarrolladores a garantizar que el sistema funciona de la forma esperada, mitigar riesgos legales, de cumplimiento, seguridad y reputación de la inteligencia artificial en producción. Incluso, estas técnicas en ocasiones son utilizadas para depurar modelos.

Al margen de lo dicho en cuanto a métodos de explicación, existen los dependientes del modelo, que son técnicas que explican las decisiones tomadas por un modelo de aprendizaje automático específico. Y por otro lado, los métodos de explicación independientes del modelo o agnósticos consideradas técnicas que pueden explicar las decisiones de cualquier modelo. Entre los métodos de explicación dependientes del modelo se incluyen LRP (del inglés, Layer-wise Relevance Propagation), Grad-CAM (del inglés, Gradient-weighted Class Activation Mapping), DeepLIFT (del inglés, Deep Learning Important FeATures), que junto a otras técnicas y métodos existentes según las características específicas del modelo que se explica, son útiles para obtener información sobre cómo se realizan predicciones y comprender la relación entre estas y los rasgos de entrada, ya que a menudo, se basan en la arquitectura y los parámetros específicos del modelo (Onose, 2023).

Por otra parte, existen numerosos métodos agnósticos y entre los más utilizados se encuentran el de la dependencia parcial (PDP, del inglés Partial Dependence Plot) como

método global y como métodos locales se destacan LIME (Local Interpretable Model-Agnostic Explanations) y SHAP (SHapley Additive exPlanations) (Molnar, 2022). En este contexto, y en correlación a los avances actuales en el campo de la XAI, existen herramientas que se utilizar para facilitar el trabajo a desarrolladores y demás personas vinculadas a este campo, como son Captum, AIX360, TFX, Scikit-learn o sklearn, Alibi, iNNvestigate, InterpretML, OmniXAI, DeepLIFT, ELI5, Skater y DALEX.

A modo general, se han propuesto varias formas de evaluar y medir la efectividad de una explicación. Cada una de ellas se establece dentro de un entorno que depende de la tarea, habilidades, y expectativas del usuario del sistema de IA, siendo estas, por lo tanto, dependientes del dominio. Algunos enfoques existentes en la explicabilidad son el análisis de sensibilidad, la descomposición simple de Taylor y las técnicas de propagación hacia atrás, sin embargo, es difícil determinar objetivamente si una técnica es buena o no, para lo cual existen estrategias de evaluación de la calidad (Montavon et al., 2018).

A pesar de los recientes avances en la XAI, la explicabilidad sigue siendo un desafío especialmente en los modelos de caja negra. Si bien existen varios métodos y marcos de trabajo, todavía se necesita más investigación y desarrollo en esta área, ya que por ejemplo, las técnicas existentes pueden proporcionar información sobre el proceso de toma de decisiones de un modelo, sin embargo a medida que estos se vuelven cada vez más complejos, se vuelve más difícil comprender cómo toman decisiones, lo que genera preocupaciones sobre su confiabilidad, usabilidad y otras pueden ser difíciles de aplicar en la práctica (Carvalho et al., 2019).

En lo que se refiere a cajas negras, un ejemplo claro y popular es la clasificación de imágenes, ya que las soluciones propuestas carecen de la capacidad de interpretar su mecanismo de trabajo interno y explicar el razonamiento principal de sus predicciones. Actualmente existen varios modelos que se pueden utilizar para la clasificación de imágenes, cada uno con sus propias fortalezas y debilidades, entre los cuales los más utilizados son ResNet, Inception, VGG y DenseNet (Carvalho et al., 2019). Otros modelos que se pueden utilizar para la clasificación de imágenes incluyen AlexNet (Anwar, 2019), GoogLeNet (Sekhar and Yang, 2022) y MobileNet(Zhang et al., 2022). Es importante tener en cuenta que el rendimiento de todos los modelos puede variar según el conjunto de datos específico y la tarea en cuestión. Por lo tanto, a menudo es necesario experimentar con diferentes modelos y ajustarlos para lograr los mejores resultados. Además, tener diversos conjuntos de datos de clasificación de imágenes es fundamental para construir modelos más precisos (Le, 2018).

En la actualidad son insuficientes los métodos de explicabilidad confiables dados las disímiles topologías de red. Por consecuente existe la necesidad de una mayor investigación y desarrollo, crucial para generar confianza, aumentar su usabilidad, comprender su proceso de toma de decisiones y reforzar el uso ético y responsable de la IA en esta área, lo cual constituye el problema de investigación de este trabajo. A partir de esto se define como objetivo general:

### **Objetivo general**

Desarrollar un método de explicabilidad para interpretar los resultados de modelos

entrenados para la clasificación de imágenes.

### **Objetivos específicos**

1. Analizar el marco teórico referencial
2. Implementar un nuevo método de explicabilidad dependiente del modelo.
3. Evaluar método propuesto en distintos casos de estudio.

Esta tesis se organiza en introducción, tres capítulos, conclusiones, recomendaciones y por último referencias bibliográficas. En el capítulo 1 se presentan los elementos teóricos para la comprensión de las redes neuronales, la explicabilidad en la IA y en especial en el área del procesamiento de imágenes, así como también se aborda acerca de los resultados obtenidos hasta la actualidad en la explicabilidad de los modelos. En el capítulo 2 se parte de la conceptualización de estudios de la explicabilidad para exponer la nueva propuesta, su diseño e implementación. En el capítulo 3 se analizan y visualizan los resultados experimentales obtenidos sintetizando los resultados obtenidos, a partir de los casos de estudio.

# **Capítulo 1**

# **Capítulo 1**

## **Aspectos Teóricos**

En este capítulo se hace un recorrido teórico para la comprensión de redes neuronales profundas, partiendo de varios conceptos generales que forman la base de la explicabilidad. Se analiza el desarrollo actual de esta y cómo se aprecian en la práctica.

### **1.1. Inteligencia Artificial**

Durante miles de años, se ha tratado de comprender cómo los humanos piensan y actúan, es decir, cómo el cerebro puede percibir, comprender, predecir y manipular un mundo mucho más grande y complicado que él mismo. El campo de la IA, se ocupa no solo de comprender, sino también de construir entidades inteligentes, máquinas que pueden calcular cómo actuar de manera efectiva y segura en una amplia variedad de situaciones novedosas. Ha evolucionado a lo largo de los años, desde los primeros intentos de crear máquinas que imitaran la inteligencia humana en la década de 1950 hasta la actualidad, en la que se utiliza en una amplia variedad de aplicaciones. Históricamente, los investigadores han buscado varias definiciones diferentes de la IA. Según (Russell and Norving, 2020) algunos han definido la inteligencia en términos de fidelidad al desempeño humano, mientras que otros prefieren la racionalidad, en cuanto a exactitud matemática. Los métodos utilizados son necesariamente diferentes: la búsqueda de una inteligencia similar a la humana debe ser en parte una ciencia empírica relacionada con la psicología, que involucra observaciones e hipótesis sobre el comportamiento humano real y los procesos de pensamiento; un enfoque racionalista, por otro lado, implica una combinación de matemáticas e ingeniería, y se conecta con la estadística, teoría del control y economía.

En (Xu et al., 2021) se refiere a la IA como la simulación de la inteligencia humana por un sistema, cuyo objetivo es desarrollar una máquina que pueda pensar como los humanos e imitar comportamientos humanos, incluyendo percepción, razonamiento, aprendizaje, planificación, predicción, y así sucesivamente. Los campos de investigación incluyen algoritmos de búsqueda, grafos de conocimiento, sistemas basados en reglas, procesamiento de lenguaje natural, razonamiento basado en casos, sistemas expertos, algoritmos de evolución, ML, DL, visión y audición artificial, etc.

En base a (Ramírez Véliz, 2022) se define la IA como una de las ramas de la Informática,

con fuertes raíces en otras áreas como la lógica y las ciencias cognitivas, y la diferenciación de sus definiciones parte de las características o propiedades que estos programas deben satisfacer, aunque todas ellas concurren en la precisión de la validación del trabajo. Dependemos de la tecnología y ésta de nosotros, la finalidad es interactuar con ella y más que competir complementarnos, así toda IA que se desarrolle permitirá hacer más de lo que podían hacer quienes antecedieron a la generación actual.

## 1.2. Aprendizaje Automático

El ML surgió en 1959 como una rama de la IA que se enfoca en el desarrollo de algoritmos que permiten a las computadoras la capacidad de aprender y mejorar a partir de la experiencia sin ser programadas explícitamente. En las últimas décadas, el ML ha experimentado un gran avance gracias a la disponibilidad de grandes cantidades de datos y la capacidad de procesamiento de las computadoras modernas. (Ongsulee, 2017). Según (Taye, 2023) este subdominio consiste en que la máquina haga juicios y pronósticos basados en datos históricos, apoyado en el uso de algoritmos para analizar datos, aprender patrones y mejorar el conocimiento y los criterios de rendimiento de los sistemas. El aprendizaje automático se utiliza en muchas áreas diferentes, que incluyen, entre otras, robótica, asistentes personales virtuales (como Google), videojuegos, reconocimiento de patrones, procesamiento de lenguaje natural, minería de datos, predicción de tráfico, redes de transporte en línea (como las estimaciones de precios de Uber), recomendaciones de productos, pronósticos del mercado de valores, diagnósticos médicos, predicciones de fraude, asesoramiento agrícola y refinamiento de resultados de motores de búsqueda (como el motor de búsqueda de Google). En fin, se materializa en diversas tareas asociadas a mecanismos específicos en dependencia del problema a resolver, que a la vez están estrechamente vinculados al tipo de aprendizaje.

De acuerdo con (Taye, 2023) el aprendizaje se clasifica en correspondencia con la forma en que un algoritmo aprende a hacer predicciones más precisas, y existen cuatro metodologías de aprendizaje fundamentales como puede ver también en la Figura 1.1:

- Aprendizaje supervisado: El algoritmo se entrena utilizando datos etiquetados. Los datos se denominan etiquetados porque consisten en pares, la salida deseada que se puede definir como una señal de supervisión y la entrada que se puede expresar como un vector, así que este algoritmo se produce cuando se conoce de antemano el resultado correcto. Con el tiempo, el algoritmo de aprendizaje refina sus resultados en un esfuerzo por reducir la brecha entre sus predicciones y el resultado real con asesoría de alguna persona especializada. Las dos subcategorías principales de aprendizaje supervisado son algoritmos de regresión (salida de valores numéricos continuos como la predicción del precio de una casa en función de su tamaño y ubicación) y algoritmos de clasificación (salida de valores discretos como la clase o categoría a la que pertenece un objeto).
- Aprendizaje no supervisado: Este enfoque entrena el algoritmo utilizando un conjunto de datos de entrada desprovisto de cualquier salida etiquetada, en contraste con el aprendizaje supervisado. Para cada elemento de entrada, no hay

una salida correcta o incorrecta, y no hay participación humana para corregir o adaptar. Por lo tanto, el aprendizaje no supervisado es más arbitrario que el supervisado. El objetivo principal de este es obtener una comprensión más profunda de los datos mediante el reconocimiento de su estructura básica o patrón de distribución. De esta forma, el algoritmo intenta representar un patrón detectado específico mientras lo refleja en la estructura general de los patrones de entrada a medida que aprende por sí mismo. Como resultado, las diversas entradas se pueden agrupar en función de las características que se tomaron de cada elemento de entrada. Es por ello que el aprendizaje no supervisado se utiliza para resolver problemas de asociación y agrupación en clústeres, para extraer características de datos sin etiquetar y categorizarlas o etiquetarlas.

- Aprendizaje semi-supervisado : utiliza una gran cantidad de datos de entrada, algunos de los cuales están etiquetados, mientras que el resto no y por tanto requiere menos interacción humana. Dado que los conjuntos de datos etiquetados son más difíciles de obtener y tal vez necesiten acceso a especialistas en dominios y los datos sin etiquetar son menos costosos y más simples de recuperar, se pueden usar enfoques de entrenamiento supervisados y no supervisados para enseñar el algoritmo en el aprendizaje semi-supervisado. Al emplear métodos de ambos aprendizajes, se pueden exponer los patrones y estructuras latentes del conjunto de datos de entrada y aplicar a datos no etiquetados para proporcionar predicciones basadas en las mejores conjeturas, que posteriormente se pueden aplicar a nuevos conjuntos de datos. Por lo tanto, se puede resumir que los datos no etiquetados se usan para volver a clasificar o reevaluar una hipótesis o pronóstico establecido usando datos etiquetados.
- Aprendizaje reforzado: en lugar de recibir instrucciones explícitas sobre qué hacer, este tipo de algoritmos aprende a través de sus propias actividades, podría caracterizarse como un proceso de aprendizaje basado en prueba y error. Se utiliza para entrenar a un agente para que tome decisiones en un entorno dinámico. Este tipo de aprendizaje utiliza un sistema de “recompensas y castigos”, ya que tienen objetivos establecidos y reciben señales que indican si una acción realizada fue exitosa o no, aspirando a la capacidad de elegir opciones que maximicen el valor de la recompensa y reduzcan la penalización. Estas técnicas se utilizan en aplicaciones como los juegos y la robótica, por ejemplo un agente en un automóvil autónomo sería recompensado por llegar a la ubicación pero castigado por desviarse de la carretera.

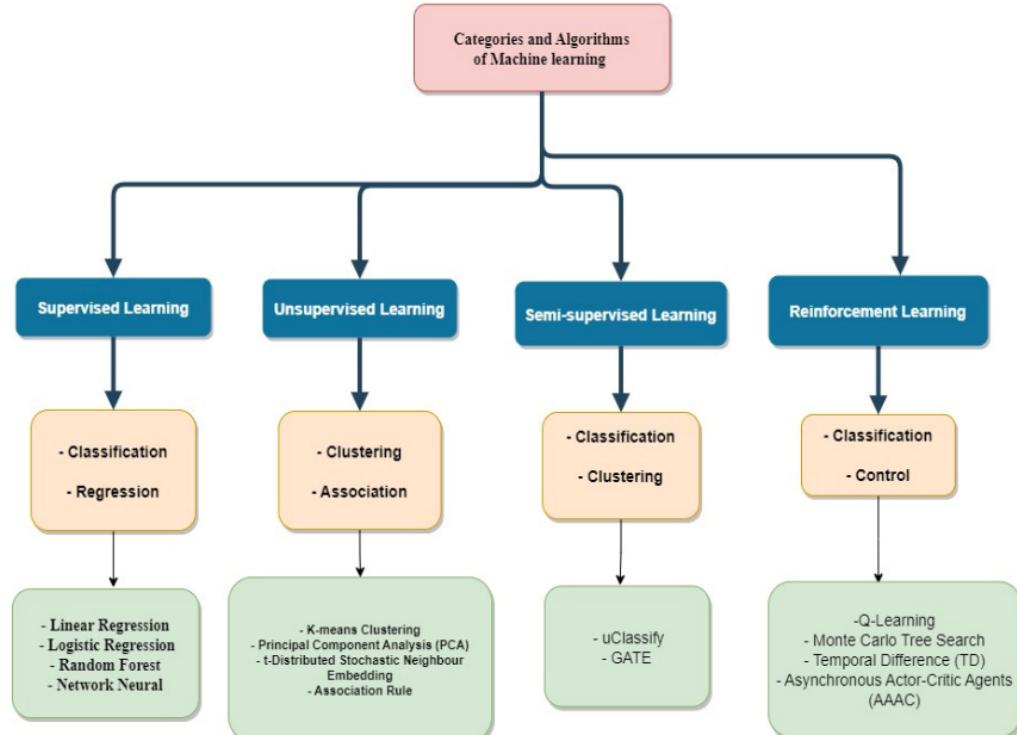


Figura 1.1: Diferentes categorías y algoritmos de aprendizaje automático. (Extraído de (Taye, 2023))

Además existen varios enfoques del aprendizaje en ML en dependencia de los datos y el problema a resolver, es importante tener en cuenta que no son mutuamente excluyentes porque se pueden combinar para mejorar el rendimiento de los modelos. A continuación, se describen algunos de estos enfoques:

- Aprendizaje Perezoso (en inglés Lazy Learning): es un enfoque en el cual el modelo pospone la generalización hasta el momento de la predicción. Algunos métodos de aprendizaje perezoso incluyen el aprendizaje basado en instancias (en inglés Instance-Based Learning) y el algoritmo del k-vecino más cercano(KNN del inglés k-nearest neighbors) (Aha and Kibler, 1991)
- Aprendizaje basado en Árboles de Decisión (en inglés Decision Tree Learning): se construye un modelo en forma de estructura de árbol donde cada nodo interno representa una característica o atributo, cada rama representa una decisión basada en esa característica, y cada hoja representa una clase o una predicción. (Tan et al., 2005)
- Aprendizaje Basado en Reglas (en inglés Rule-Based Learning): se centra en extraer reglas o condiciones “if-then” a partir de los datos de entrenamiento. Estas reglas se utilizan posteriormente para hacer predicciones o tomar decisiones. (Zhou and Purvis, 2004)

- Aprendizaje Bayesiano (en inglés Bayesian Learning): utiliza el teorema de Bayes para actualizar las creencias o conocimientos previos a medida que se obtienen nuevos datos. Este enfoque se basa en la inferencia estadística y permite la estimación de la probabilidad de eventos futuros.(Barber, 2012)
- Aprendizaje por Transferencia (en inglés Transfer Learning): implica aprovechar el conocimiento o la experiencia adquirida en una tarea o dominio para mejorar el rendimiento en un dominio objetivo relacionado.(Tan et al., 2018)
- Aprendizaje Conexionista (en inglés Connectionist Learning): según (Bishop, 2006) se basa en redes neuronales artificiales , las que se abordaran con mayor profundidad en el próximo apartado.

### 1.3. Redes Neuronales

Las Redes Neuronales Artificiales, (ANN del inglés Artificial Neural Networks o NN de Neural Networks) surgieron con el propósito de procesar información y resolver problemas en diversos campos, están inspiradas en las redes neuronales biológicas del cerebro humano y están constituidas por elementos que se comportan de forma similar a la neurona biológica en sus funciones más comunes. Según (Basogain Olabe, 2003) estos elementos están organizados de una forma parecida a la que presenta el cerebro humano, donde la unidad análoga a la neurona biológica es el elemento procesador, llamado neurona artificial. Esta tiene varias entradas con un peso asociado y las combina, normalmente con una suma básica. La suma de las entradas es modificada por una función de transferencia o activación y el valor de la salida de esta función de se pasa directamente a la salida de la neurona. Además esta se puede conectar a las entradas de otras neuronas artificiales mediante conexiones ponderadas correspondientes a la eficacia de la sinapsis (la señal) de las conexiones neuronales. A continuación en la figura 1.2 se presenta la ilustración de una neurona artificial y sus componentes.

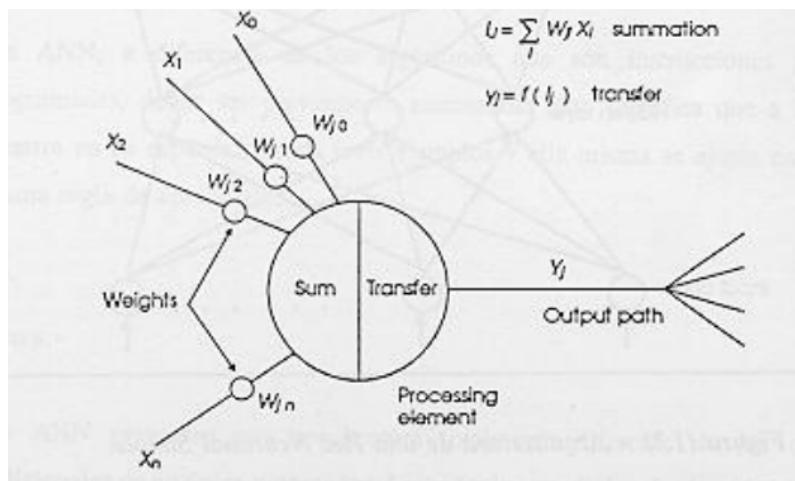


Figura 1.2: Diagrama de una neurona artificial. (Extraído de (Basogain Olabe, 2003))

Las ANN al margen de “parecerse” al cerebro presentan una serie de características propias del cerebro, por ejemplo las ANN aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos. Sin embargo en (Amador, 2023) se expone que poseen otras funcionalidades y estructuras de conexión distintas a las vistas desde la perspectiva biológica. Entre sus características principales están las siguientes:

1. Auto-organización y adaptabilidad: utilizan algoritmos de aprendizaje adaptativo y auto-organización, por lo que ofrecen mejores posibilidades de procesado robusto y adaptativo.
2. Procesado no lineal: aumenta la capacidad de la red para aproximar funciones, clasificar patrones y aumenta su inmunidad frente al ruido.
3. Procesado paralelo: normalmente se usa un gran número de nodos de procesado, con alto nivel de interconectividad.

### 1.3.1. Fases de modelación con redes neuronales

Las fases de modelar y construir NNs pueden variar dependiendo de la aplicación específica y el tipo de red que se está modelando. Sin embargo, una generalización del modelado según (Wang et al., 2021; Haykin, 2009; Zhang and Zhang, 2019) serían:

1. Recopilación y preprocesamiento de datos: se recopilan y preprocesan los datos necesarios para entrenar y probar la NN. Esto incluye la limpieza de los datos, la eliminación de valores atípicos y la normalización de los datos para garantizar que sean adecuados para su uso en la red neuronal.
2. Selección de modelo y diseño de arquitectura de red: se selecciona el tipo de red neuronal a utilizar, y se diseña su arquitectura. Esto incluye determinar el número de capas, el número de neuronas en cada capa y las funciones de activación que se utilizarán.
3. Entrenamiento: la red neuronal se entrena utilizando los datos preprocesados. Este proceso implica ajustar los pesos y sesgos de la red para minimizar el error entre la salida predicha y la salida real.
4. Validación: el rendimiento de la red neuronal entrenada se evalúa utilizando un conjunto de datos de validación. Esto ayuda a garantizar que la red neuronal no se ajuste en exceso a los datos de entrenamiento y pueda generalizarse bien a nuevos datos, previendo los problemas de sobreajuste y subajuste.
5. Prueba: el rendimiento de la red se evalúa utilizando un conjunto de datos de prueba separado. Esto ayuda a garantizar que la red neuronal pueda predecir con precisión las salidas de datos nuevos.
6. Despliegue: la red neuronal entrenada se despliega en una aplicación del mundo real. Esto implica integrar la red neuronal en la aplicación y asegurarse de que funciona correctamente.

En ocasiones, las fases de validación y prueba se realiza en una única fase combinada, dependiendo del tipo de red y tareas específicas encomendadas.

### 1.3.2. Topología de redes neuronales

En el desarrollo de una NN, no hay que programar ni el conocimiento ni las reglas del procesamiento del conocimiento. La red aprende las reglas del procesamiento del conocimiento mediante el ajuste de las conexiones ponderadas entre las neuronas de distintas capas de la red, dado que, la mayoría de ANN se caracterizan por tener sus neuronas divididas en capas, comenzando por la capa de entrada, por la que se recibirá la señal entrante de la red, y terminando con la capa de salida por donde la red enviará su señal de salida, pudiendo o no existir entre ellas, capas intermedias denominadas capas ocultas, como se ve en la figura 1.3 (Moreno, 2020). Al margen de lo dicho, la topología o arquitectura de una ANN hace referencia a cómo están estructuradas las capas y las conexiones.

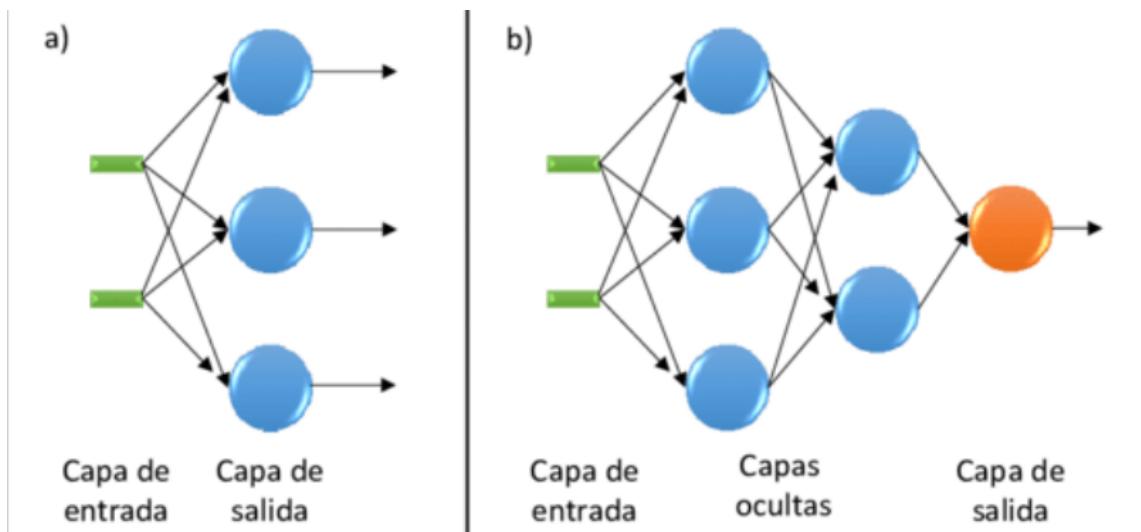


Figura 1.3: a) Red neuronal simple o monocapa (en inglés Single-layer Perceptron). b) Red neuronal multicapa (MLP del inglés Multi-layer Perceptron). (Extraído de (Manjarrez, 2014))

Las conexiones entre neuronas son un aspecto crucial, ya que determinan el flujo de información a través de la red. Pueden ser totales o parciales, dependiendo de la fuerza de la conexión. Las conexiones totales significan que todas las neuronas en una capa están conectadas a todas las neuronas en la siguiente capa, mientras que las conexiones parciales significan que solo algunas neuronas están conectadas. Además, para (de Boves Harrington et al., 2002) entre dos capas de neuronas diferentes existen conexiones hacia delante, hacia atrás, lateral y de retardo, ver figura 1.4:

- Conexiones hacia delante (en inglés feedforward): la información fluye en una dirección, desde la capa de entrada a la capa de salida. La capa de entrada recibe

información, la procesa y la pasa a la siguiente capa hasta que llega a la capa de salida.

- Conexiones hacia atrás (en inglés feedback): en esta estructura, la información fluye en la dirección opuesta, desde la capa de salida a la capa de entrada. La capa de salida produce un resultado, que luego se compara con la salida deseada, y el error se propaga de vuelta a través de la red para ajustar los pesos de las conexiones.
- Conexiones laterales: en esta, la información fluye entre neuronas en la misma capa. Esto permite el procesamiento de información en paralelo, lo que puede ser útil para tareas como el reconocimiento de patrones.
- Conexiones retardadas: acá la información se almacena en una neurona durante un cierto período de tiempo antes de transmitirse a la siguiente neurona. Esto puede ser útil para tareas donde la red necesita recordar información específica procesada anteriormente.

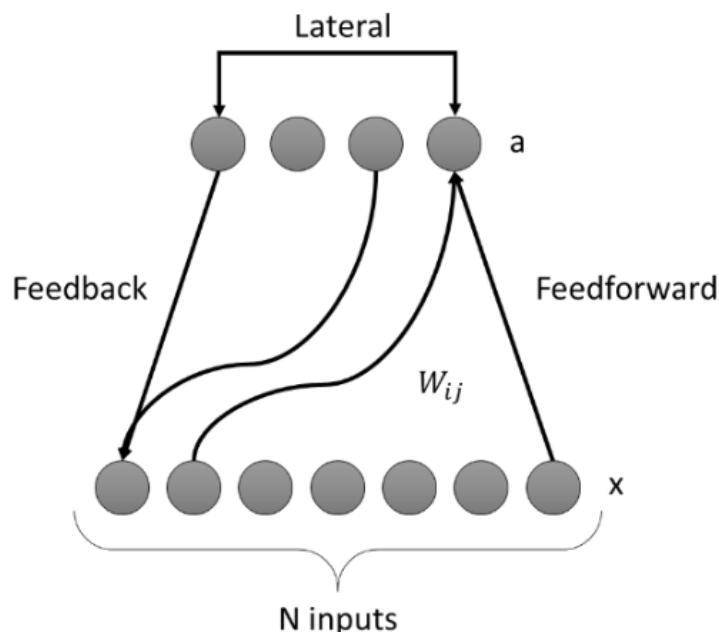


Figura 1.4: Conexiones de una red neuronal. (Extraído de (de Boves Harrington et al., 2002))

El número de capas ocultas y el número de neuronas en cada capa son factores importantes para determinar además el rendimiento de una ANN. En este sentido, han sido estudiadas varias topologías de redes a partir de las conexiones de sus capas, como es la red neuronal de alimentación directa(FFN del inglés Feedforward neural network), donde la información fluye en una sola dirección a través de conexiones feedforward, a

su vez se suele entrenar por medio del método backpropagation para obtener los pesos de las conexiones internas durante el entrenamiento, de forma que calcula el error entre las salidas predichas por la red y las salidas deseadas, y luego lo propaga hacia atrás a través de la red para ajustar los pesos y minimizar el error, ver el esquema de funcionamiento de una DNN en la figura 1.5 a continuación. (Zhang et al., 2021)

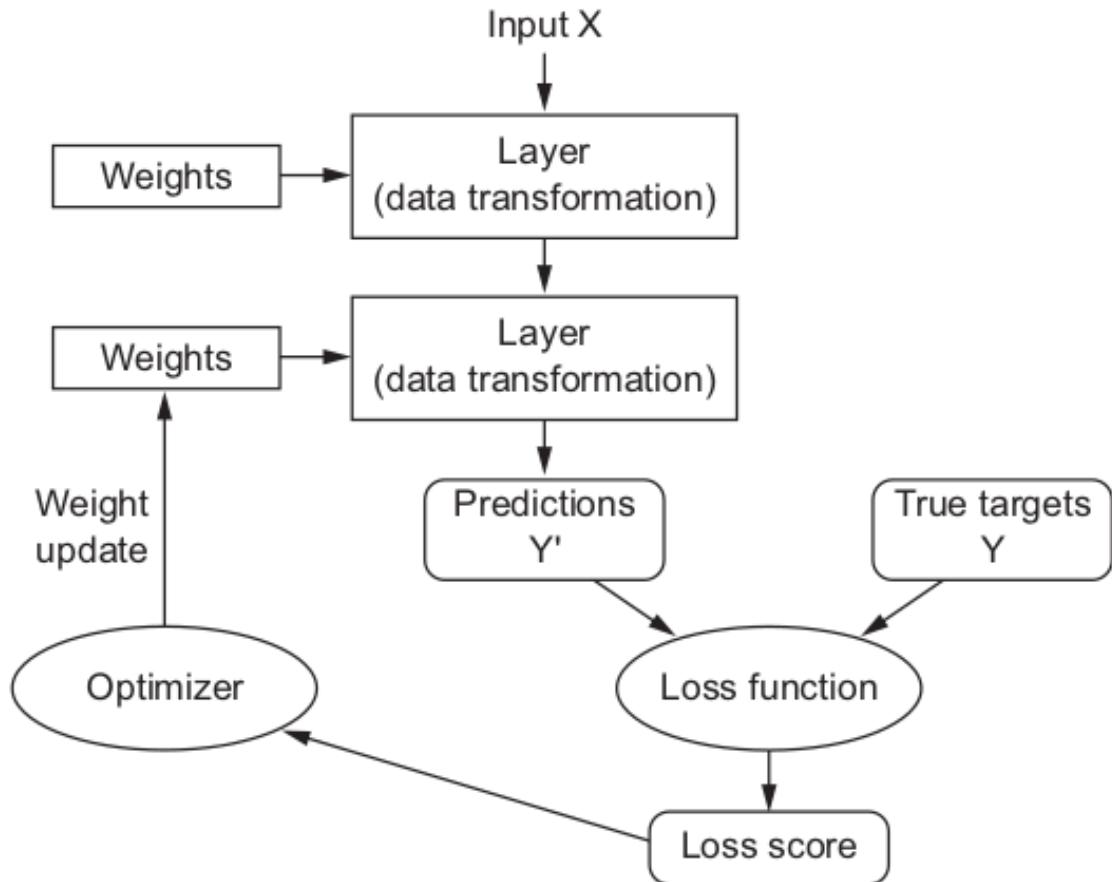


Figura 1.5: Proceso de entrenamiento de una red neuronal. (Extraído de (Chollet, 2021))

En correlación, un MLP, visto anteriormente en la figura 1.3 b), es un tipo de FFN que tiene una o más capas ocultas, que constituye la arquitectura básica de las redes neuronales profundas.

## 1.4. Aprendizaje Profundo

El aprendizaje profundo constituye una de las áreas de investigación más popular dentro del campo de la IA, y es un subcampo dentro del aprendizaje automático, que se utiliza

para resolver problemas muy complejos que normalmente implican grandes cantidades de datos. Se caracteriza, por su profundidad y por su capacidad para aprender las representaciones de los datos. Utiliza distintas estructuras de redes neuronales para lograr el aprendizaje de sucesivas capas de representaciones cada vez más significativas de los datos. En un modelo en general se suelen utilizar decenas o incluso cientos de capas de representación las cuales aprenden automáticamente a medida que el modelo es entrenado con los datos. (Díaz-Ramírez, 2021)

El surgimiento de esta área según (Sarker, 2021) se debe a varios factores, en primer lugar, los avances en el poder de cómputo y el acceso a grandes conjuntos de datos, lo que ha permitido entrenar redes en este subcampo de manera más eficiente. Además, con su aparición y evolución se han logrado avances significativos en diferentes campos y un rendimiento sobresaliente en tareas como el reconocimiento de imágenes, el procesamiento del lenguaje natural, la detección de fraudes, la conducción autónoma, la asistencia médica y la traducción automática, entre otros.

### 1.4.1. Redes Neuronales Profundas

Una red neuronal profunda( DNN, del inglés Deep Neural Network) es una RNA con varias capas ocultas entre las capas de entrada y salida que pueden modelar relaciones no lineales complejas. Se utilizan ampliamente en el aprendizaje supervisado y en los problemas de aprendizaje por refuerzo. (Díaz-Ramírez, 2021)

Las múltiples capas ocultas presentes en las DNNs son las que les permiten aprender características y patrones más complejos de los datos. El término “profundo” se refiere al número de capas en la red, y la profundidad de la red es uno de los factores que contribuyen a su rendimiento. En cuanto a las MLPs se consideran NN poco profundas en ocasiones, cuando tienen una o dos capas ocultas solamente, mientras las DNN tienen mucho más de dos capas ocultas como se ve en la figura 1.6. La decisión de utilizar un MLP o un DNN depende de la complejidad de la tarea y de la cantidad de datos disponibles. Para tareas más simples con conjuntos de datos más pequeños, un MLP puede ser suficiente y más eficiente desde el punto de vista computacional. Sin embargo, para tareas más complejas con conjuntos de datos más grandes, puede ser necesario una DNN para lograr un mejor rendimiento aunque costoso computacionalmente. (Kriegeskorte and Golan, 2019)

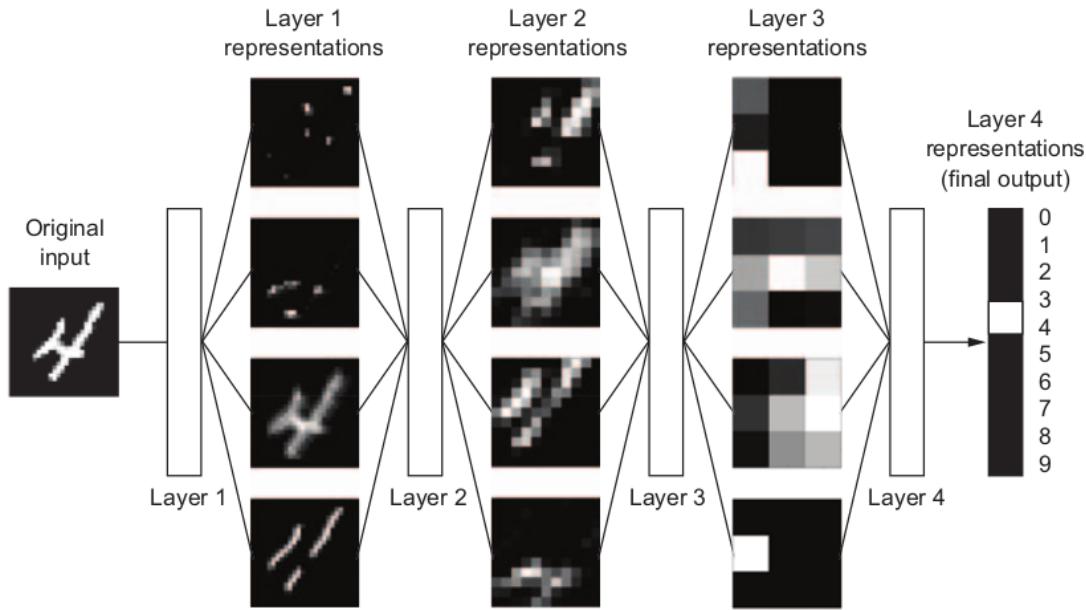


Figura 1.6: Representación del aprendizaje profundo para un modelo de clasificación de dígitos. (Extraído de (Chollet, 2021))

El esquema básico de su estructura cuenta con las siguientes capas, ver figura 1.7:

- **Capa de entrada:** Se corresponde con el vector de datos de entrada. Si se dispone de un vector  $x$  de datos, cada elemento del vector  $(x_1, x_2, \dots, x_n)$  constituye una neurona de entrada.
- **Capa de salida:** Es la capa en la que se realiza la operación objetivo, por ejemplo, la clasificación. En ese caso, la capa de salida tiene tantas neuronas como clases presente el problema a tratar. Cada neurona tiene asociado un peso y un bias, en español conocido como sesgo, puede referirse a la tendencia de un modelo a favorecer ciertos patrones sobre otros debido a la arquitectura o al proceso de optimización.
- **Capas ocultas:** Son las que contienen todos los cálculos intermedios de la red. Están formadas por neuronas ocultas, cada una de las cuales tiene un peso y un bias, al igual que las neuronas de la capa de salida.

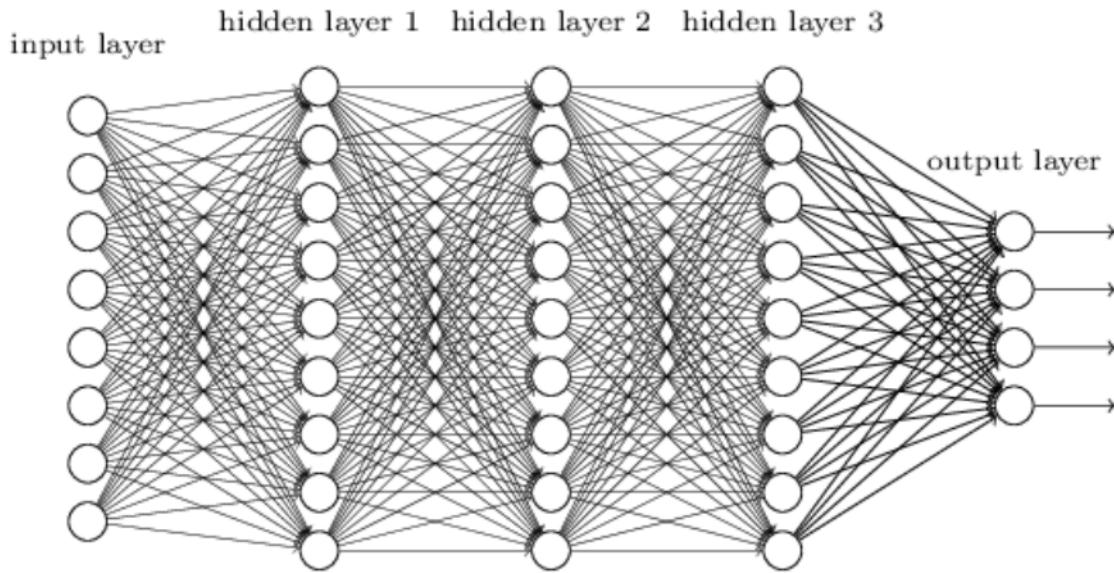


Figura 1.7: Red neuronal profunda. (Extraído de (Kilpi, 2017))

### 1.4.2. Redes Neuronales Profundas Convolucionales

Existen muchos ejemplos o tipos de redes neuronales profundas, y cada uno tiene sus propias fortalezas y debilidades, que se adapta a diferentes tipos de tareas y datos. Las redes neuronales convolucionales (CNNs del inglés Convolutional Neural Networks) son un tipo de arquitectura de redes neuronales profundas que se inspira en el procesamiento visual del cerebro humano. Los algoritmos tradicionales de aprendizaje no pudieron extraer de manera efectiva características significativas de las imágenes, así es que se diseñaron las CNN para superar esta limitación mediante el AA de representaciones jerárquicas de datos de imagen, que pueden capturar características de bajo y alto nivel. La convolución se refiere a la operación matemática de aplicar un conjunto de filtros (pequeñas matrices de ponderaciones) que se aprenden durante el proceso de entrenamiento, a la imagen de entrada, para producir un conjunto de mapas de características que capturan diferentes aspectos de la imagen de entrada. (Lanjewar and Gurav, 2022; Gong et al., 2019)

El esquema básico de las CNN consta de varias capas que trabajan juntas para extraer características de las imágenes de entrada y realizar la tarea. El siguiente es un desglose de las capas en una CNN típica, y en la figura 1.8 se visualiza de forma resumida:

- Capa de entrada: toma la imagen de entrada y la pasa a la siguiente capa.
- Capa convolucional: la operación de convolución se realiza multiplicando los valores de filtro por los valores de píxel correspondientes en la imagen y sumando

los resultados para producir un único valor de salida. Este proceso se repite para cada ubicación en la imagen de entrada, produciendo un conjunto de mapas de características.

- Capa de activación: aplica una función de activación a la salida de la capa convolucional como ReLu, para introducir no linealidad en el modelo.
- Capa de agrupación o submuestreo(en inglés pooling): reduce la resolución de los mapas de características para reducir su dimensionalidad espacial al tiempo que conserva la información más importante.
- Capa oculta o completamente conectada (en inglés fully-connected): toma la salida de la capa de agrupación y la alimenta a una o más capas completamente conectadas, que realizan la tarea especificada, donde la primera capa se encargará de convertir la matriz de datos en un vector plano, es por ello que esta capa suele recibir el nombre de capa plana (en inglés flatten).
- Capa de abandono ( en inglés Dropout) es una máscara que anula la contribución de algunas neuronas hacia la siguiente capa y deja sin modificar todas las demás. Podemos aplicar una capa Dropout al vector de entrada, en cuyo caso anula algunas de sus características; pero también podemos aplicarlo a una capa oculta, en cuyo caso anula algunas neuronas ocultas.
- Capa de salida: produce la salida final del modelo, que puede ser una distribución de probabilidad sobre las clases posibles en el caso de clasificación de imágenes.

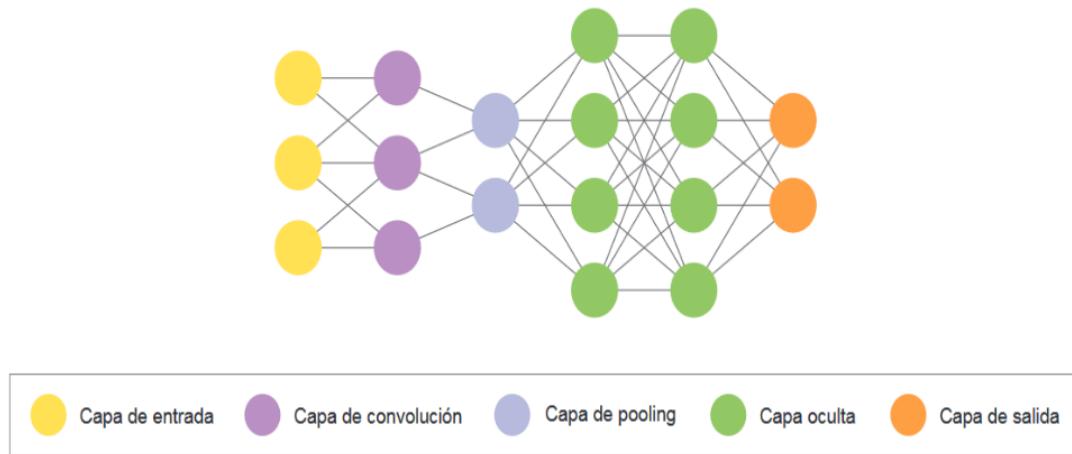


Figura 1.8: Red neuronal convolucional. (Extraído de (Amador, 2023))

Una primera capa de convolución aprenderá pequeños patrones locales, como bordes, una segunda capa de convolución aprenderá patrones más grandes hechos de las características

de las primeras capas, y así sucesivamente. Esto permite a los CNN aprender de manera eficiente conceptos visuales cada vez más complejos y abstractos (porque el mundo visual es fundamentalmente jerárquico espacialmente). Ver figura 1.9

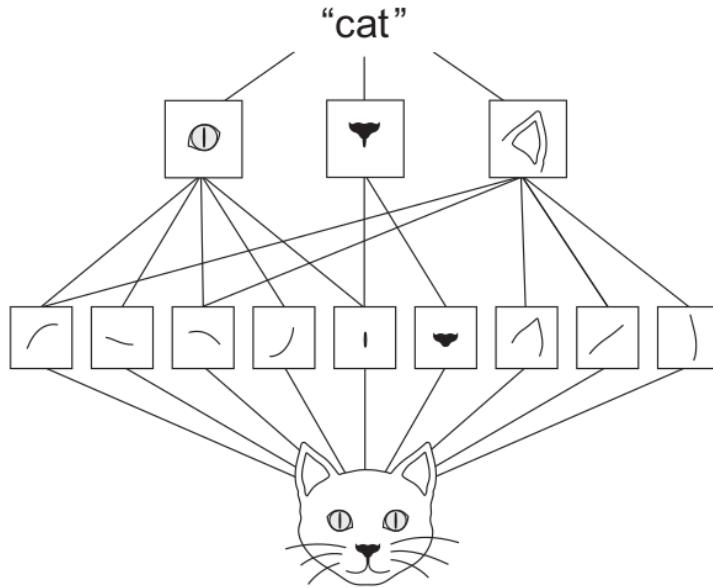


Figura 1.9: El mundo visual forma una jerarquía espacial de lo visual módulos: los bordes hiperlocales se combinan en objetos locales como ojos u orejas, que se combinan en conceptos de alto nivel como “gato.” (Extraído de (Chollet, 2021))

Las capas de una CNN desglosadas anteriormente se pueden repetir varias veces para crear una arquitectura de red más profunda. En la figura 1.10 puede apreciarse el esquema de una CNN para el ejemplo de clasificación de una imagen. La función ReLu es la más utilizada debido a que permite el aprendizaje muy rápido en las RNA. Si a esta función se le da valores de entrada muy negativos el resultado es cero pero si se le da valores positivos queda igual. La función de activación softmax transforma las salidas sin procesar de la RNA en un vector de probabilidades, esencialmente una distribución de probabilidad sobre las clases de entrada. Considere un problema de clasificación multiclase con  $n$  clases. La activación softmax devuelve un vector de salida que tiene  $n$  entradas, con la entrada en el índice  $i$  correspondiente a la probabilidad de que una entrada particular pertenezca a la clase  $i$ , como se puede ver en la imagen a continuación.

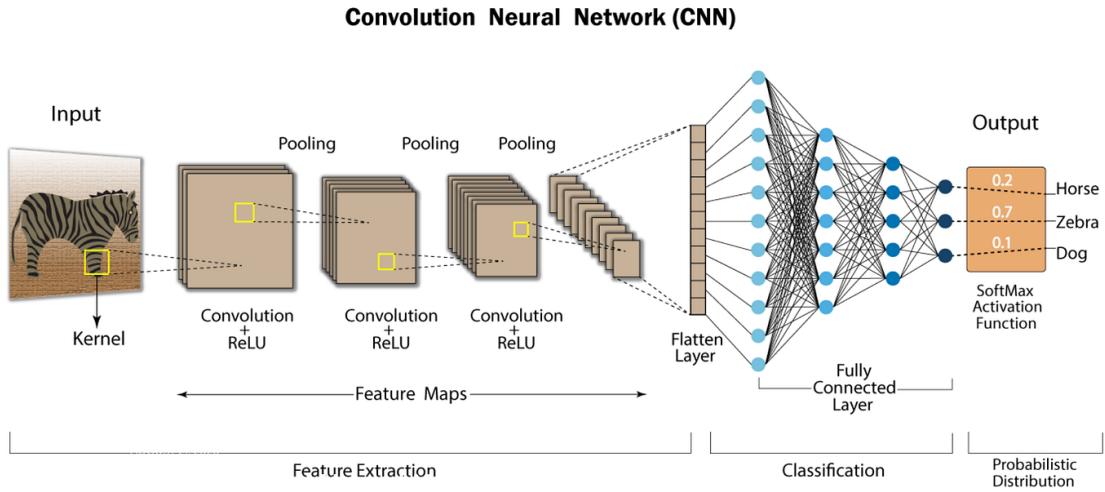


Figura 1.10: Diagrama de capas de una red neuronal convolucional en la clasificación de imágenes. (Extraído de (Modi, 2023))

La arquitectura de una CNN puede variar dependiendo del problema que se intenta resolver y normalmente se optimiza mediante el uso de hiperparámetros, como el número de capas ocultas, el número de neuronas en cada capa, la elección de la función de activación, la elección del método de optimización y las técnicas de regularización. (Wang and Zhang, 2023; Gandikota, 2019)

Las CNN son similares a las redes MLP, su principal diferencia radica en la inclusión de capas convolucionales, cuyas neuronas no están totalmente conectadas: cada neurona de una capa no recibe conexiones entrantes de todas las neuronas de la capa anterior, sino sólo de algunas, lo cual favorece que cada neurona se especialice únicamente en una región de la capa anterior, reduciendo drásticamente el número de operaciones a realizar. De esta forma, las redes convolucionales dividen y modelan la información en partes más pequeñas, para combinar después esta información en las capas más profundas de la red. Ver figura 1.11 (Alzubaidi et al., 2021)

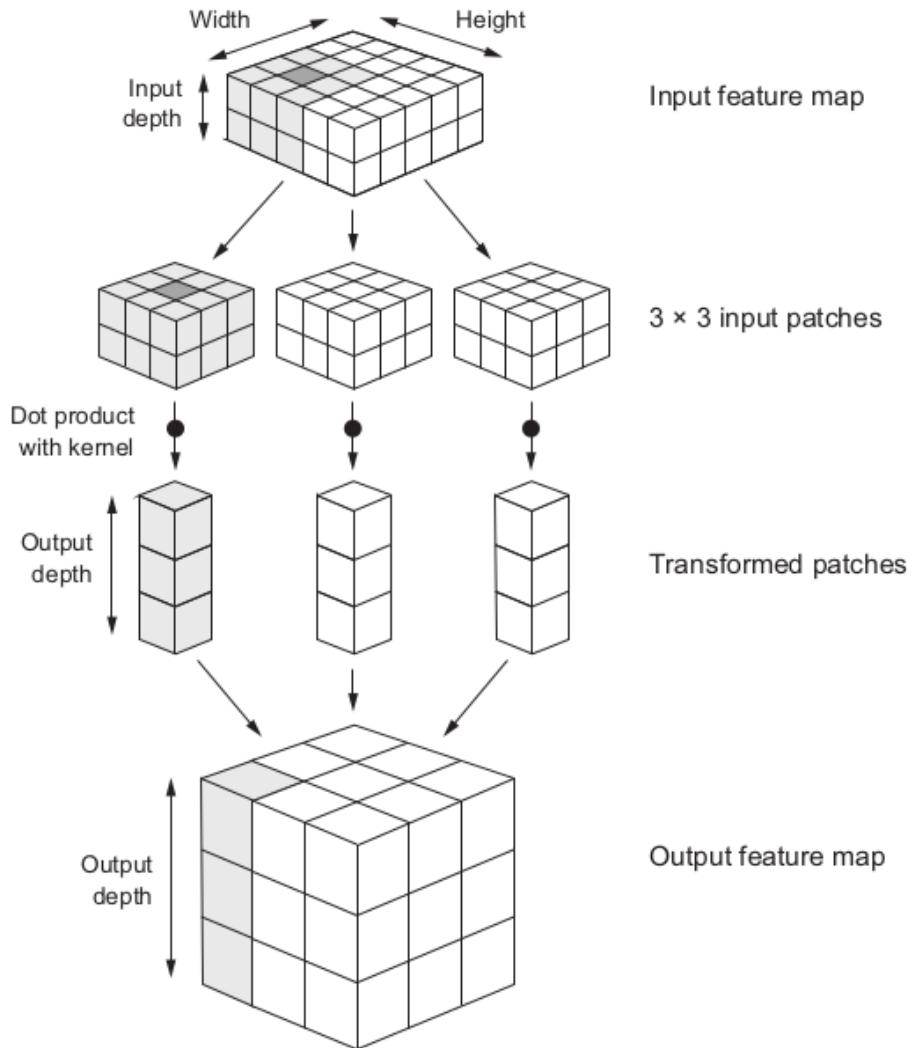


Figura 1.11: Diagrama del proceso de convolución.( Extraído de (Chollet, 2021))

Un modelo de DNN, transforma sus datos de entrada en resultados significativos, un proceso que se “aprende” de la exposición a ejemplos conocidos de entradas y salidas. Por lo tanto, para (Chollet, 2021) el problema central en el ML y el DL es transformar los datos de manera significativa: en otras palabras, aprender representaciones útiles de los datos de entrada disponibles, que nos acercan al resultado esperado. Una representación, en esencia, es una forma diferente de ver los datos: representarlos o codificarlos. Por ejemplo, una imagen en color puede codificarse en el formato RGB (rojo-verde-azul)o en el formato HSV( tono-saturación-valor): estas son dos representaciones diferentes de los mismos datos. Algunas tareas que pueden ser difíciles con una representación pueden volverse fáciles con otra. P or ejemplo, la tarea “seleccionar todos los píxeles rojos en la

“imagen” es más simple en el formato RGB, mientras que “hacer que la imagen esté menos saturada” es más simple en el formato HSV. Los modelos de CNN tratan de encontrar representaciones apropiadas para sus datos de entrada, transformaciones de los datos que los hacen más susceptibles a la tarea en cuestión, como una tarea de clasificación.

### 1.4.3. Modelos de redes convolucionales

Actualmente existen varios modelos que se pueden utilizar para la clasificación de imágenes, cada uno con sus particularidades, de los cuales a continuación se ejemplifica uno de ellos.

**VGG** para (Zielinski et al., 2020) es un modelo que se introdujo en 2014 que consiste en una serie de capas convolucionales seguidas de capas completamente conectadas, diseñado para lograr una alta precisión en las tareas de clasificación de imágenes, aunque es computacionalmente costoso debido a su gran cantidad de parámetros. El modelo VGG tiene 16 o 19 capas, dependiendo de la variante utilizada. VGG16 tiene 13 capas convolucionales y 3 capas completamente conectadas, mientras que VGG19 tiene 16 capas convolucionales y 3 capas completamente conectadas. Ver diagrama del modelo en la figura 1.12

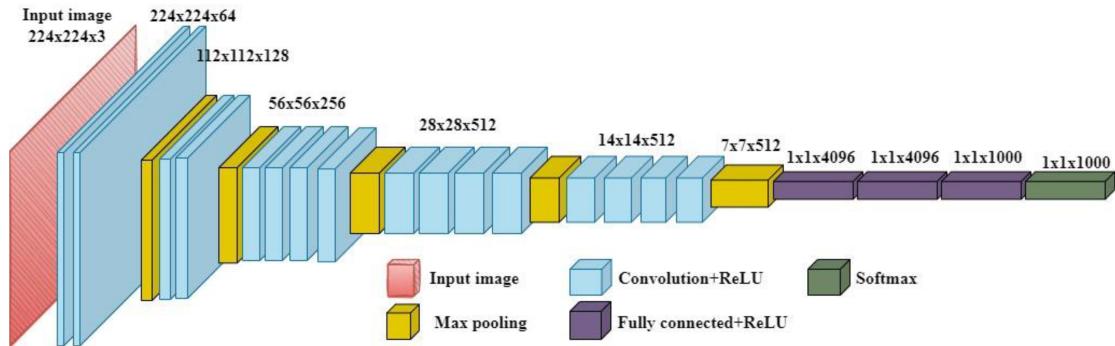


Figura 1.12: Diagrama de capas de VGG19. Extraído de (Nguyen et al., 2022)

En general, los modelos existentes son una herramienta importante en la clasificación de imágenes, ya que proporcionan un punto de partida para un entrenamiento eficiente, una precisión y solidez mejorada. Un enfoque común y altamente efectivo para el DL en pequeños conjuntos de datos de imágenes es para usar una red previamente entrenada. Una red pre-entrenada es una red guardada que se entrenó anteriormente en un conjunto de datos grande, generalmente en una tarea de clasificación de imágenes a gran escala, como son los modelos mencionados anteriormente. Si este conjunto de datos original es lo suficientemente grande y general, entonces la jerarquía espacial de características aprendidas por la red puede actuar efectivamente como un modelo genérico del mundo

visual y, por lo tanto, sus características pueden resultar útiles para muchos problemas de visión por computadora diferentes, aunque estos nuevos problemas puedan involucrar clases completamente diferentes a las de la tarea original.

## 1.5. Inteligencia Artificial Explicable

A pesar que existen modelos interpretables como los árboles de decisión o los modelos basados en reglas, existen otros como los mencionados anteriormente en el campo del DL, que aunque alcanzan precisiones de predicción impresionantes, su estructura no lineal anidada los hace altamente no transparentes, es decir, no está claro qué información en los datos de entrada los hace llegar realmente a sus decisiones. Ver figura 1.13 Por lo que, estos modelos se consideran típicamente como cajas negras(en inglés, black-boxes), mientras que los modelos diseñados para ser interpretables son los de cajas blancas(en inglés, white-boxes) (Samek et al., 2019)

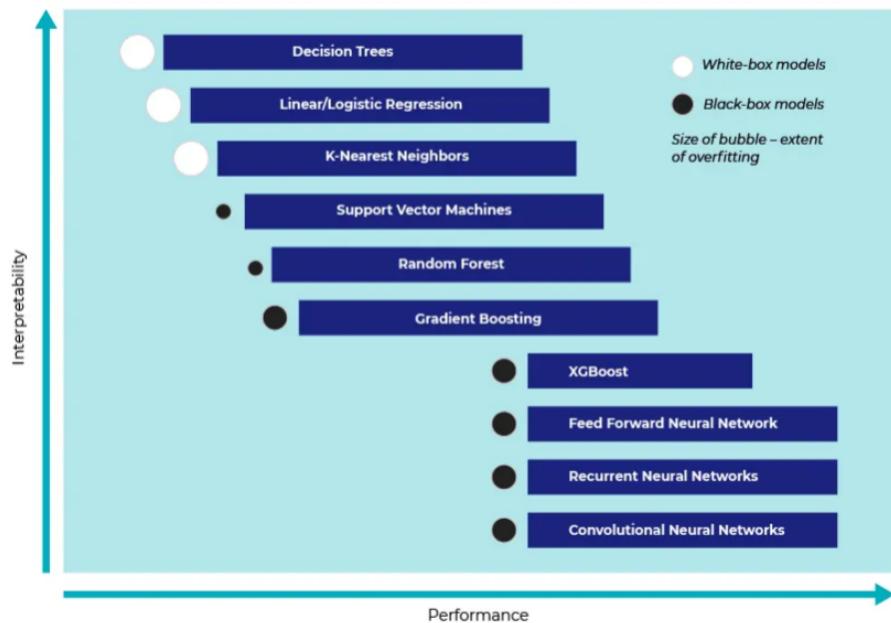


Figura 1.13: Modelos de caja negra y modelos de caja blanca. (Extraído de (Badalia, 2021))

Para abordar este problema, los investigadores han desarrollado técnicas para hacer que los modelos de caja negra sean más interpretables y transparentes. Estas técnicas se han desarrollado en un área de investigación propia, conocida como XAI. Para (Molnar, 2022) la XAI es un enfoque del ML que tiene como objetivo proporcionar información sobre cómo un modelo toma decisiones, haciendo que este proceso sea transparente e interpretable, lo que puede ayudar a los usuarios a comprender y confiar en las predicciones del modelo. Las técnicas en la XAI incluyen métodos para visualizar el proceso de toma de decisiones de los modelos, identificar características o entradas

importantes y generar explicaciones para las predicciones de los modelos. Además se pueden utilizar para identificar posibles sesgos o errores en un modelo, lo que puede ayudar a mejorar la precisión y la imparcialidad de este.

La XAI es importante y la humanidad necesita de ella por varias razones según (Samek et al., 2019):

- Verificación del sistema: es necesario comprender el proceso de toma de decisiones de un sistema de IA para garantizar su precisión y confiabilidad. Por ejemplo, en el diagnóstico médico, sería irresponsable confiar en las predicciones de un sistema de caja negra por defecto. En cambio, cada decisión de gran alcance debe ser accesible para una validación adecuada por parte de un experto humano.
- Mejora del sistema: comprender las debilidades de un sistema es el primer paso para mejorarlo. Es más fácil realizar este análisis de debilidades en modelos interpretables que en modelos de caja negra. Además, detectar sesgos en el modelo o en el conjunto de datos es más fácil si se entiende lo que el modelo está haciendo y por qué llega a sus predicciones.
- Aprendizaje del sistema: los sistemas de IA se entrena con millones de ejemplos, y pueden observar patrones en los datos que no son accesibles para los humanos. Al utilizar sistemas de XAI, podemos intentar extraer este conocimiento destilado del sistema para adquirir nuevos conocimientos.
- Cumplimiento de la legislación: las IAs están afectando cada vez más áreas de nuestra vida diaria, y los aspectos legales, como la asignación de responsabilidad cuando el sistema toma una decisión equivocada, han recibido recientemente una mayor atención. Dado que puede ser imposible encontrar respuestas satisfactorias para estas preguntas legales cuando se depende de modelos de caja negra, los futuros sistemas de IA necesariamente tendrán que volverse más explicables.

En resumen, la XAI es importante para garantizar la precisión y confiabilidad de los sistemas de IA, mejorarlo, aprender de ellos y sin dejar de lado la ética legal. Ver figura 1.14

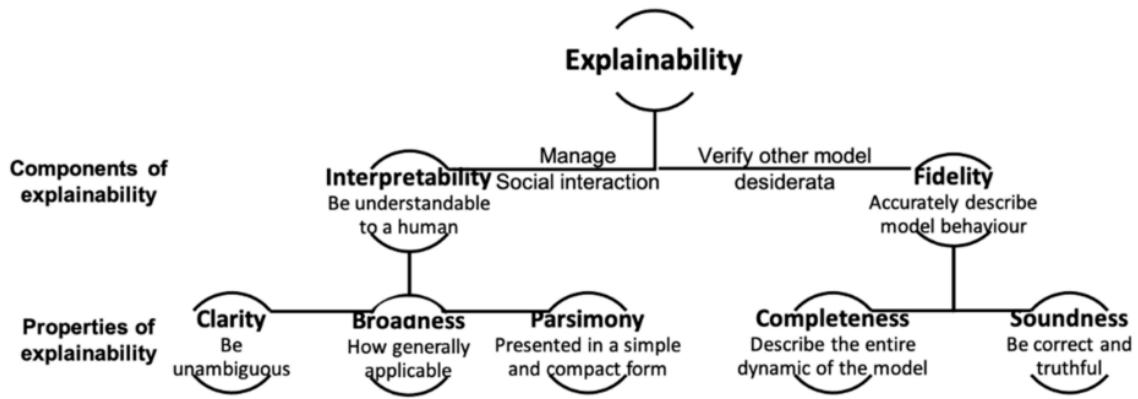


Figura 1.14: Definición de explicabilidad y propiedades relacionadas. (Extraído de (Zhou et al., 2021))

### 1.5.1. Explicabilidad

En la XAI se han propuesto diferentes tipos de explicabilidad basados en los enfoques de generación de explicaciones, ver figura 1.15 el tipo de explicación, el alcance de la explicación, el tipo de modelo que puede explicar o combinaciones de estos métodos. Por ejemplo, al considerar cuándo son aplicables las explicaciones, los métodos de explicación se pueden agrupar en premodelo (ante-hoc), en modelo, y métodos posteriores al modelo(post-hoc), los dos primeros se centran en la utilización de un modelo, o en la incorporación de algoritmos al proceso de diseño o entrenamiento, que sean intrínsecamente interpretables, es decir fáciles de entender y explicar sin necesidad de técnicas adicionales, como los árboles de decisión. Mientras que la explicabilidad post-hoc se refiere a la capacidad de explicar cómo un modelo de ML tomó una decisión después de que se haya entrenado y se haya utilizado para hacer predicciones.(Zhou et al., 2021)

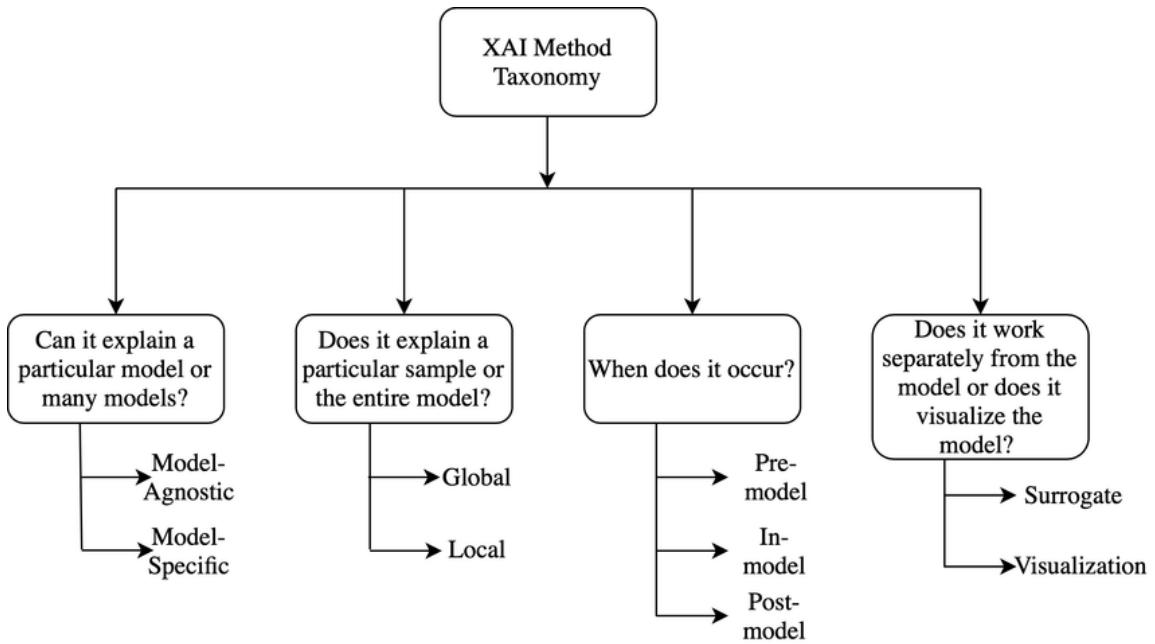


Figura 1.15: Enfoques de los métodos de explicabilidad. (Extraído de (Zhou et al., 2021))

También hay métodos específicos del modelo y agnósticos o independientes del modelo, así como métodos de explicación globales y locales como se ve en la figura 1.16. Para (Molnar, 2019) las explicaciones globales buscan comprender el modelo en su conjunto, identificando las variables más importantes y cómo se relacionan entre sí, para determinar toda la lógica de un modelo y el razonamiento detrás de todos los resultados posibles. Y por otro lado los métodos locales se centran en comprender el modelo en un punto de datos específico, identificando qué variables contribuyen a la predicción y cómo afectan los valores de esas variables a la predicción, para datos o rasgos específicos.

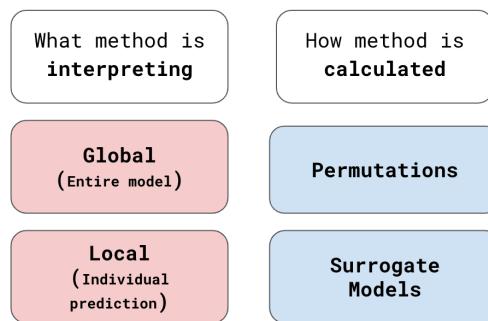


Figura 1.16: Métodos de explicabilidad. (Extraído de (O'Sullivan, 2022))

Por lo que respecta a los métodos dependientes o no del modelo, en (Onose, 2023) los

métodos dependientes son técnicas de explicabilidad que se aplican a un modelo de IA específico, teniendo en cuenta su arquitectura y parámetros. Proporcionan información detallada sobre cómo funciona un modelo en particular y están diseñados para ser utilizados en el contexto de ese modelo específico. Por el contrario, los agnósticos se pueden aplicar a cualquier modelo de IA, independientemente de su arquitectura o parámetros. Proporcionan información y no están vinculados a características específicas de un modelo en particular. En (Arya et al., 2019) se presentan numerosos ejemplos de métodos existentes teniendo en cuenta entre otros enfoques las definiciones anteriores, donde algunos de los más utilizados son:

### **Métodos específicos del modelo**

- LRP: método basado en la propagación hacia atrás de la relevancia, que es una técnica que se utiliza para retroceder a través de las capas de un modelo y asignar relevancia a las características de entrada.
- Grad-CAM: método utilizado para visualizar las áreas de una imagen que son más importantes para la predicción del modelo. Este utiliza la información de gradiente de una CNN para generar un mapa de calor que muestra las áreas de la imagen que son más importantes para la predicción del modelo.
- DeepLIFT: método usado para asignar relevancia a las características de entrada de un modelo. Se basa en la idea de que la relevancia de un rasgo de entrada se puede calcular comparando la activación de este en una entrada con la activación promedio de la característica en todas las entradas.

### **Ejemplos de métodos agnósticos del modelo**

- SHAP: es un método de explicación local y post-hoc que asigna un valor de importancia a cada característica en el modelo basado en su contribución a la predicción, se basa en la teoría de Shapley (marco matemático para asignar la contribución de cada jugador en un juego cooperativo) y utiliza una aproximación de Monte Carlo para calcular la contribución de cada característica en el modelo.
- LIME: método de explicación local y post-hoc que aproxima el modelo con un modelo más simple e interpretable en la vecindad de la predicción perturbando las características de entrada y observando cómo cambia la salida.
- PDP: método de explicación global y post-hoc que se utiliza para analizar la relación entre una variable de entrada y la salida del modelo, mientras se mantienen constantes los valores de las demás variables de entrada. Esto lo logra mediante la creación de un gráfico que muestra cómo cambia la salida del modelo a medida que se varía el valor, lo que permite identificar patrones y tendencias en los datos.

Es importante tener en cuenta que diferentes tipos de explicaciones, pueden ser más apropiados para diferentes tareas y contextos, y no existe un enfoque único para la explicabilidad en el aprendizaje automático. Por lo tanto, es necesario evaluar y comparar

diferentes métodos de explicación para determinar cuál es el más adecuado para una tarea específica.

### 1.5.2. Herramientas actuales para la explicabilidad

En (Chollet, 2021) uno de los factores clave que impulsa el desarrollo del aprendizaje profundo ha sido la democratización de los conjuntos de herramientas utilizados en el campo, en los primeros tiempos para trabajar en esta área se requería una gran experiencia en C++ y CUDA, que pocas personas poseían. Hoy en día, las habilidades básicas de scripting de Python son suficientes para realizar investigaciones avanzadas. Esto ha sido impulsado principalmente por el desarrollo de Theano, luego TensorFlow y posteriormente Pytorch, tres marcos simbólicos de manipulación de tensores para Python que admiten la diferenciación automática, lo que simplifica enormemente la implementación de nuevos modelos, y por el surgimiento de bibliotecas fáciles de usar como Scikit-learn y más tarde Keras. Después de los lanzamientos anteriores, entre otras herramientas, se ha logrado que sea tan fácil como manipular ladrillos LEGO, convirtiéndose estas herramientas en la solución de aprendizaje profundo para grandes número de nuevas empresas, estudiantes de posgrado e investigadores que giran en el campo.

En correlación a lo antes expuesto, existen herramientas que se pueden utilizar para facilitar el trabajo a desarrolladores y demás personas vinculadas a este campo, tal cual se explica en (Molnar, 2022; Onose, 2023; Amesoeder et al., 2023; Apley and Zhu, 2019; Alber et al., 2019) como son las herramientas y bibliotecas mencionadas a continuación:

- LIME, es una biblioteca de python que proporciona explicaciones respecto a las predicciones de cualquier clasificador aproximándolo localmente con un modelo interpretable. Para usar este método, debe proporcionar una función que pueda predecir la salida para cualquier entrada dada, y una función que pueda calcular la distancia entre dos entradas cualesquiera. Ver documentación e instalación en <https://pypi.org/project/lime/>
- SHAP, es una biblioteca de python que proporciona explicaciones, de forma que calcula la contribución de cada característica a la predicción de un modelo. Es importante tener en cuenta que para obtener resultados significativos, los datos deben limpiarse, procesarse previamente y estar listos para el modelado. Además, es una buena práctica evaluar el desempeño del modelo antes de interpretar los resultados de las explicaciones. Ver más acerca de su instalación y documentación disponibles en <https://shap.readthedocs.io/en/latest/>
- AIX360 (AI Explainability 360), es una biblioteca de código abierto de IBM para python, para la interpretabilidad y la explicabilidad de conjuntos de datos y modelos de ML. Incluye una colección de algoritmos que cubren diferentes dimensiones de explicaciones junto con métricas de explicabilidad. Ver acerca de su instalación y documentación en <https://aix360.readthedocs.io/en/latest/>
- Captum de Pytorch: es una biblioteca de código abierto de PyTorch para la interpretación de modelos que proporciona varios algoritmos y admite la mayoría

de los tipos de modelos de PyTorch con una modificación mínima de la red neuronal. Ver documentación disponible en <https://captum.ai/>

- OmniXAI, es una biblioteca de python que ofrece XAI omnidireccional y capacidades de ML interpretables para abordar muchos puntos débiles al explicar las decisiones tomadas por los modelos en la práctica. Proporciona una interfaz fácil de usar para realizar XAI. Esta pretende ser una biblioteca integral que facilite la explicación de la IA para los científicos de datos, investigadores de ML y profesionales que necesitan explicaciones en su proyecto. La última versión incluye un explicador GPT experimental. Este explicador aprovecha los resultados producidos por SHARP y MACE para formular el mensaje de entrada para ChatGPT. Posteriormente, ChatGPT analiza estos resultados y genera las explicaciones correspondientes que proporcionan a los desarrolladores una comprensión más clara de la justificación detrás de las predicciones del modelo. Ver documentación completa disponible en <https://opensource.salesforce.com/OmniXAI/latest/index.html> o <https://pypi.org/project/omnixai/>
- DeepLIFT (Deep Learning Important FeaTures), es un paquete de Python que profundiza en la selección de características de una red neuronal y encuentra neuronas y pesos que tuvieron efectos importantes en la formación de resultados. Ver documentación disponible en <https://pypi.org/project/deeplift/>
- Skater, es un marco unificado de Python de código abierto e independiente del modelo para la explicabilidad e interpretabilidad. Tiene tanto métodos globales como locales. Ver más acerca en su documentación disponible en <https://pypi.org/project/skater/>
- ELI5 (Explain Like I'm Five), es una biblioteca de Python que proporciona funciones para visualizar y comprender las predicciones de varios tipos de modelos, incluidos modelos lineales, árboles de decisión y modelos de caja negra como Random Forest, XGBoost y redes neuronales. Ver acerca de su instalación y documentación en <https://eli5.readthedocs.io/en/latest/overview.html>
- BreakDown, es una biblioteca de python para explicar las predicciones de los modelos de aprendizaje automático, particularmente en el contexto de los modelos lineales. Proporciona explicaciones interpretables de las predicciones del modelo al descomponer la predicción en las contribuciones de cada característica, de una manera que los humanos puedan entender fácilmente. Ver documentación disponible en <https://pypi.org/project/breakdown/>
- Alibi, es una biblioteca de python de código abierto para inspección e interpretación de modelos. Proporciona el código necesario para producir explicaciones para algoritmos de caja negra. El objetivo de la biblioteca es proporcionar implementaciones de alta calidad de métodos de explicación de caja

negra, caja blanca, locales y globales para modelos de clasificación y regresión. Documentación disponible en <https://pypi.org/project/alibi/>

- iNNvestigate, es una herramienta para el lenguaje de programación Python para investigar las predicciones de las redes neuronales. Se basa en Keras y TensorFlow 2.0. Proporciona una implementación lista para usar numerosos métodos de análisis. Ver más acerca de esta biblioteca, en su documentación oficial disponible en <https://pypi.org/project/innvestigate/>
- Dalex, es un paquete de python que examina cualquier modelo dado, simple o complejo, y explica el comportamiento del modelo. Crea un nivel de abstracción alrededor de cada modelo que hace que sea más fácil de explorar y explicar. Este paquete hace una especie de “radiografía” cualquier modelo y ayuda a explorar y explicar su comportamiento, para comprender cómo funcionan los modelos complejos. El objeto explicador principal crea un contenedor alrededor de un modelo predictivo. Los modelos envueltos se pueden explorar y comparar con una colección de explicaciones a nivel de modelo y a nivel de predicción. Además, hay métodos de equidad y paneles de exploración interactivos disponibles para el usuario. Ver más acerca en su documentación <https://pypi.org/project/dalex/>
- InterpretML, es un conjunto de herramientas de código abierto desarrollado por Microsoft, destinado a mejorar la explicabilidad del modelo, ofreciendo explicaciones tanto globales como parciales. Es flexible y personalizable. Ver documentación en <https://interpret.ml/docs/index.html>
- Anchors, es una biblioteca de Python para generar explicaciones interpretables por humanos para las predicciones de modelos de aprendizaje automático de caja negra. Se basa en el concepto de “anclajes”, que son un conjunto de condiciones mínimas y suficientes que un punto de datos debe satisfacer para ser clasificado como una determinada clase por un modelo de caja negra. Los anclajes se pueden usar para comprender por qué un modelo de caja negra realizó una predicción específica para un punto de datos dado. También se pueden usar para identificar posibles problemas con el modelo, como sesgos o injusticias. Ver documentación disponible en <https://pypi.org/project/anchors/>

Existen varias formas de evaluar y medir la efectividad de una explicación en el ámbito de la inteligencia artificial. Cada una de ellas se establece dentro de un entorno que depende de la tarea, habilidades y expectativas del usuario del sistema de IA, siendo estas, por lo tanto, dependientes del dominio. Algunos enfoques existentes en la explicabilidad son el análisis de sensibilidad, la descomposición simple de Taylor y las técnicas de propagación hacia atrás (Samek et al., 2019; Molnar, 2022). Sin embargo, es difícil determinar objetivamente si una técnica es buena o no, para lo cual existen estrategias de evaluación de la calidad. Es importante destacar que la explicabilidad de los modelos de IA es fundamental para comprender cómo se toman las decisiones y para detectar sesgos o errores en el proceso. Por lo tanto, es necesario seguir investigando y desarrollando nuevas técnicas y herramientas para mejorar la interpretación de los

modelos y garantizar su transparencia y responsabilidad en diferentes ámbitos, incluyendo el laboral y el público. (Cotino Hueso and Castellanos Claramunt, 2023)

A pesar de los avances en el campo de la explicabilidad en los modelos de inteligencia artificial, todavía existen desafíos importantes que deben abordarse. En particular, a medida que los modelos se vuelven cada vez más complejos, se vuelve más difícil comprender cómo toman decisiones, lo que genera preocupaciones sobre su confiabilidad y usabilidad.

## 1.6. Conclusiones parciales

El avance de las investigaciones en el campo de aprendizaje automático ha tenido gran impacto en el desarrollo de algoritmos complejos y modelos que permiten a las máquinas aprender y tomar decisiones basadas en los datos. Sin embargo, estos modelos a menudo se consideran “cajas negras”, ya que su lógica interna y sus procesos de toma de decisiones no son fácilmente interpretables por los humanos.

La falta de profundidad y transparencia en los modelos puede ser problemática, por lo que existe la necesidad de desarrollar métodos robustos y prácticos para explicar las decisiones tomadas por estos. Los métodos de explicabilidad se han vuelto fundamentales en el intento de determinar cuáles variables contribuyen y cómo afectan sus valores a las predicciones.

En el caso de los modelos de clasificación de imágenes es igualmente necesario lograr mejorar su interpretabilidad y es todo un desafío identificar cuáles son los píxeles determinantes que contribuyen al modelo para realizar la predicción de la clase resultante.

## **Capítulo 2**

# Capítulo 2

## Implementación del nuevo método

En este capítulo se presentará el método diseñado para interpretar los resultados de clasificación de imágenes tomando como ejemplo el modelo VGG19 y un conjunto de datos seleccionado. Se explican los detalles de cada uno de los pasos de procesamiento que componen el flujo y se describen los conceptos utilizados en cada etapa. Además se hace un análisis de las particularidades a tener en cuenta para el desarrollo de cada etapa, así como de los datos, la estructura de las clases y funciones utilizadas.

### 2.1. Descripción general del modelo

Se diseñó una red en cascada compuesta por dos redes, siendo la primera la encargada de la explicabilidad y la segunda el modelo escogido y entrenado para la clasificación de imágenes que se desea explicar. El método de explicabilidad diseñado está constituido por una primera red cuya implementación y detalles se abordan en los próximos apartados, y una segunda red a partir de un modelo pre-entrenado para clasificar imágenes, de forma que su trabajo en conjunto está orientado al aprendizaje de los píxeles relevantes para la decisión tomada por este último. A continuación la figura 1.16 muestra el diagrama de flujo de todo el proceso.

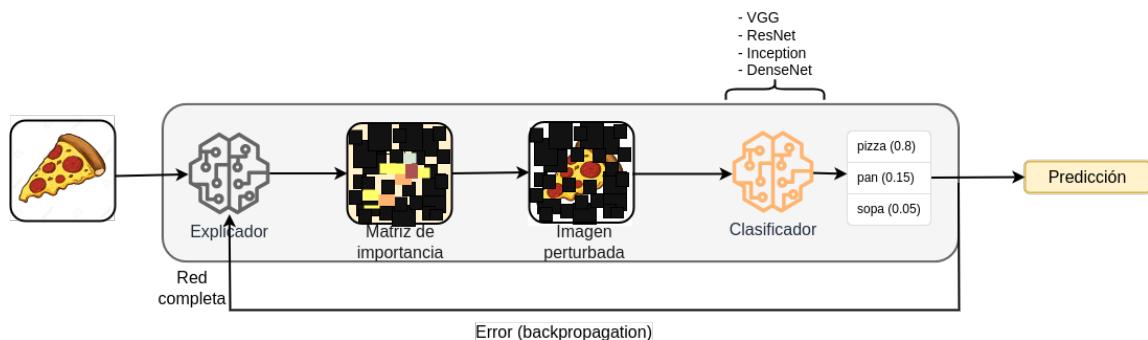


Figura 2.1: Diagrama de flujo

Visión general de los pasos a seguir en la figura 2.2

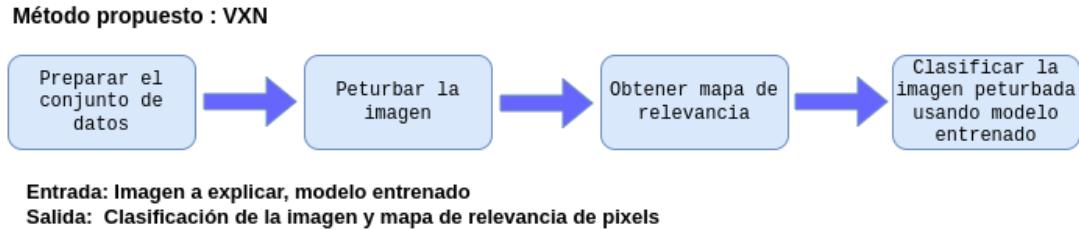


Figura 2.2: Esquema de pasos a seguir

Guía de pasos del algoritmo propuesto:

1. Preparar conjunto de datos de entrada.
  - Preprocesar las imágenes del conjunto seleccionado acorde a los datos que acepta el clasificador escogido.
  - Dividir en conjuntos de entrenamiento, validación y prueba.
  - Modificar capas de salida del clasificador de ser necesario, en dependencia de la cantidad de clases que se desea predecir.
  - Entrenar el clasificador con las imágenes preprocesadas, haciendo uso de la técnica de transferencia de aprendizaje (conocida en inglés como Transfer Learning).
  - Evaluar todos los conjuntos y eliminar las imágenes cuya predicción no fue correcta, de esta forma se garantiza el aprendizaje del explicador sin sesgo.
  - Guardar los datos preparados, el cual es un paso opcional pero recomendable, ya que los datos transformados y organizados que se almacenan, evitan la preparación en pruebas futuras.
2. Aplicar el método de explicabilidad a las imágenes.
  - Pasar cada imagen a través de las capas de la red implementada, y la salida final será la imagen original superpuesta a su *mapa de relevancia*<sup>1</sup>, resultando una imagen perturbada en dependencia de los *pesos de cada píxel*<sup>2</sup>.
  - Obtener el mapa de relevancia, es esencial para la explicación, pero debido a la lógica interna del método no es lo adquirido directamente en la salida de la red,

<sup>1</sup>El mapa de relevancia de una imagen se refiere a la representación visual de los valores de los pesos de sus píxeles.

<sup>2</sup>Los pesos de los píxeles se refieren a valores numéricos asignados a cada píxel que indican su importancia o contribución a la predicción, lo que significa que los píxeles con pesos más altos tienen una mayor influencia en la decisión final.

por lo cual se define una función que captura la salida por capas y guarda toda la información de cada etapa significativa del procesamiento de la imagen a través de la red.

### 3. Aplicar el clasificador a las imágenes perturbadas obtenidas en la salida anterior.

- Utilizar el modelo entrenado para evaluar las imágenes, congelando sus pesos para que no se actualicen en el proceso de aprendizaje y a consecuencia de esto la salida del clasificador influirá solamente en los pesos del explicador y por tanto en la interpretación de este sobre la relevancia de los píxeles para cada predicción.

#### 2.1.1. Arquitectura de la red y particularidades a tener en cuenta en cada capa

La arquitectura de la red está constituida por dos capas lineales además de la de perturbación, como se muestra a continuación en la figura 2.3

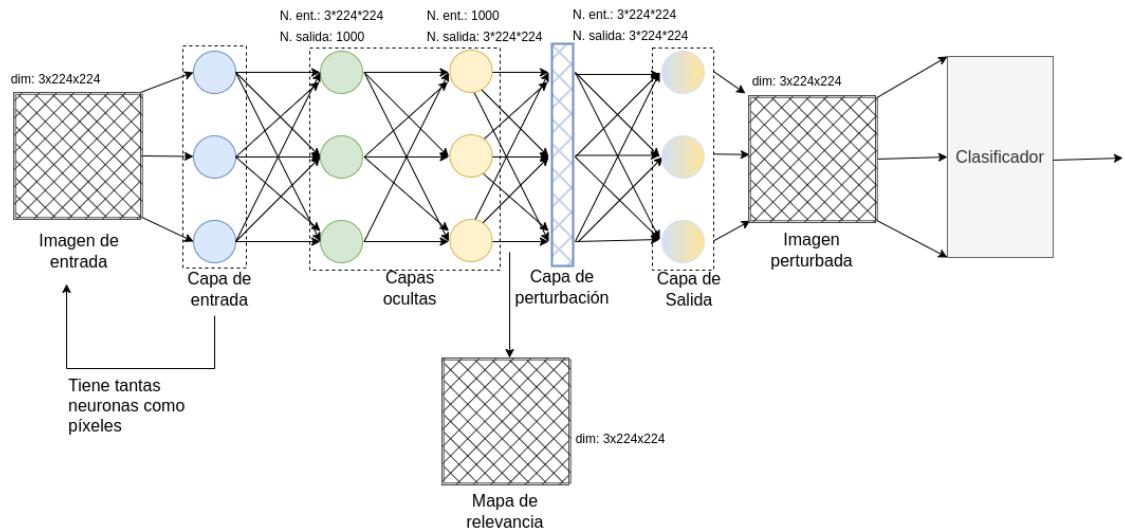


Figura 2.3: Diagrama de la arquitectura de la red

La imagen de entrada tiene una dimensión de 3x224x224, lo que significa que tiene 224 píxeles de largo y de ancho, y 3 canales de color. Esta es la dimensión utilizada para los casos de estudio específicos donde se utilice como modelo pre-entrenado VGG19, ya que es la configuración que espera este, debido a su entrenamiento y definición previa. Por consiguiente, la capa de entrada de la red tendrá tantas neuronas como píxeles tenga la imagen, recibirá la imagen en su formato equivalente a un *tensor*<sup>3</sup> tridimensional y la aplana a unidimensional para poder pasársela a la siguiente capa.

<sup>3</sup>Un tensor es una estructura de datos que generaliza conceptos como escalares, vectores y matrices a un número arbitrario de dimensiones.

La primera *capa lineal*<sup>4</sup> tiene como entrada el tensor unidimensional, por tanto debe tener tantas neuronas de entrada como resulte el tamaño de este, y de salida las definidas, en este caso son 1000. En el método *forward*<sup>5</sup> de la red, se le aplica a esta capa la *función de activación*<sup>6</sup> *ReLU*<sup>7</sup>.

La segunda capa tiene como entrada la misma cantidad de neuronas que la salida de la anterior, y de salida tantas neuronas como el tamaño del tensor aplanado de la imagen. En el *forward* se le aplica la función *Sigmoid*<sup>8</sup>, obteniendo los valores de probabilidades asociados a las relevancias de los píxeles para la clases determinadas.

A continuación se perturba la imagen original a través de la función definida  $x * y$ , donde  $x$  es el vector que almacena los valores de los píxeles de la imagen de entrada, guardado antes de pasar la imagen aplanada por las capas de la red, mientras  $y$  es el tensor de probabilidades obtenidos en la última capa. Con este paso se espera lograr que en el tensor resultante de la operación, los píxeles más relevantes tomen valores superiores a los demás y cada vez sean más acertados a los que son importantes para la predicción del clasificador. Al calcular los nuevos valores de los píxeles, se llevan nuevamente a un tensor de dimensión tridimensional para que la salida de la red sea la imagen perturbada con la misma dimensión que la de entrada, la cual será lo que reciba el clasificador, este evaluará su entrada y devolverá una salida correspondiente a la predicción. Es entonces que la red hace el Backpropagation, actualizando los pesos del explicador según los resultados de la evaluación del modelo entrenado, mientras dicho modelo mantiene sus pesos congelados ya que sólo se necesita su evaluación no su entrenamiento nuevamente. Importante aclarar que para que así sea, al definir el optimizador, solo se le debe pasar los parámetros de la primera red.

## 2.2. Implementación

Para la implementación del método se utilizaron las bibliotecas de python Pytorch con varios de sus módulos, Matplotlib, Numpy, tqdm y google.colab.drive como parte de las bibliotecas preinstaladas de Google Colab, ya que como entorno de ejecución se utiliza la GPU T4 que ofrece en su versión gratuita limitada. Ver figura 2.4

---

<sup>4</sup>La capa lineal o completamente conectada(fc del inglés fully-connected), es una capa que se encuentra entre la entrada y la salida de la red donde cada neurona está conectada a todas las neuronas de la capa anterior.

<sup>5</sup>El método *forward* de una RNA se utiliza para realizar la propagación hacia adelante de los datos a través de la red

<sup>6</sup>Una función de activación es una función matemática que se aplica a la salida de las neuronas en una capa de una RNA, a la cual permite modelar relaciones no lineales en los datos

<sup>7</sup>ReLU: función de activación definida como  $f(x) = \max(0, x)$

<sup>8</sup>Sigmoid: función de activación definida como  $f(x) = \frac{1}{1+e^{-x}}$ , produciendo una salida en el rango entre 0 y 1

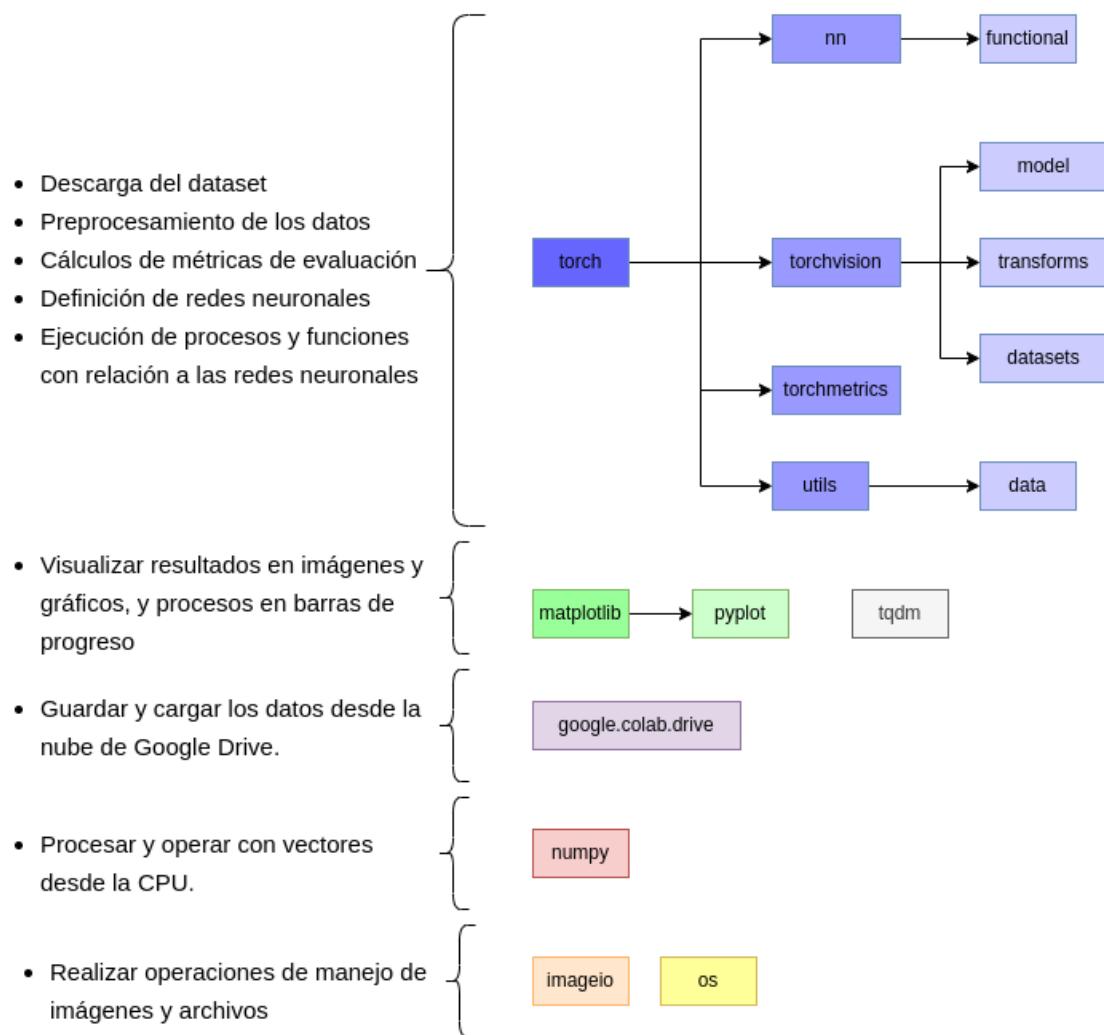


Figura 2.4: Diagrama de bibliotecas y dependencias utilizadas

A continuación, se muestra el código de partes esenciales de la implementación: las clases donde se definen las redes 2.5, las funciones definidas para visualizar 2.6 y guardar los resultados 2.7.

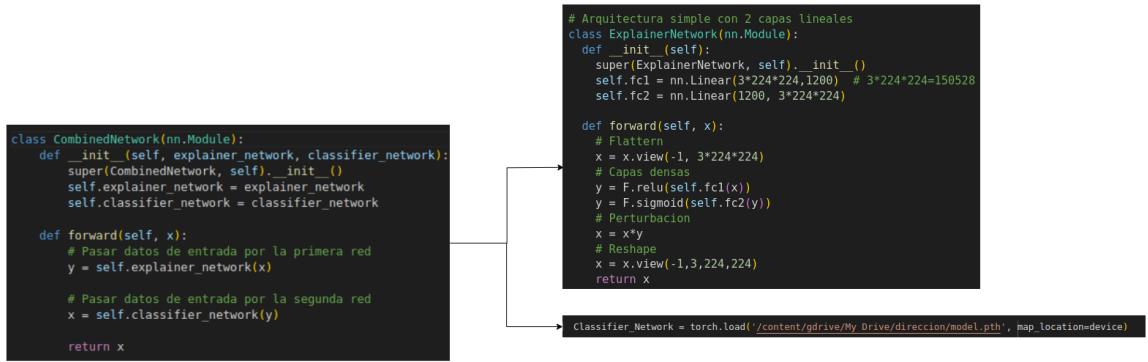


Figura 2.5: Clases de las redes implementadas

Donde *ExplainerNetwork* es la red que aprende a interpretar los resultados del clasificador entrenado y guardado *Classifier\_Network*. Ambas redes instanciadas son las que recibe como parámetros *CombinedNetwork*, la cual enlaza la salida del explicador como entrada del clasificador y será la que posteriormente se entrena para obtener las explicaciones deseadas.

```

def show_images(images,message):
    # Crea una figura con subplots
    fig, axs = plt.subplots(1, len(images), figsize=(10, 10))

    # Muestra cada imagen en un subplot diferente
    for i, (image, name) in enumerate(images):
        image=image.cpu().detach().numpy()
        image = np.transpose(image, (1, 2, 0))
        axs[i].imshow(image)
        axs[i].axis('off')
        axs[i].set_title(name)
    axs[-1].text(0, image.shape[1] + 10, message, fontsize=10, ha='right')

    # Muestra los subplots
    plt.show()

```

Figura 2.6: Función para visualizar las imágenes y su información

```

def save(combined_network,loader):
    combined_network.eval()

    # Lista para almacenar los resultados por cada imagen
    resultados = []

    # Iterar sobre el conjunto de datos de validación
    with torch.no_grad(): # Asegurarse de que no estamos realizando el seguimiento de gradientes
        for images, labels in loader:
            images, labels = images.to(device), labels.to(device)
            # Pasar las imágenes a través de la red explicadora (ExplainerNetwork)
            exp_fc1 = combined_network.explainer_network.fc1(images.view(-1, 3*224*224))
            exp_fc2 = F.sigmoid(combined_network.explainer_network.fc2(exp_fc1))
            # exp_fc2 = exp_fc2.view(-1,3,224,224)
            exp_output = combined_network.explainer_network(images.view(-1, 3*224*224))

            # Obtener la predicción del modelo clasificador
            pred = combined_network.classifier_network(images)

            # Almacenar los resultados en el diccionario
            for i in range(len(labels)):
                resultado = {
                    'Etiqueta': labels[i],
                    'Imagen_Original': images[i],
                    'Mapa_Relevancia': exp_fc2[i],
                    'Imagen_Perturbada': exp_output[i],
                    'Prediccion': torch.argmax(pred[i]).item()
                }
                resultados.append(resultado)

    return resultados

```

Figura 2.7: Función para salvar los resultados luego de entrenado el modelo

### 2.2.1. Análisis de la complejidad computacional

La complejidad computacional se refiere a cuántos recursos computacionales, como tiempo y memoria, consume un algoritmo en relación con el tamaño de la entrada. A continuación se analiza la complejidad computacional en cada parte del algoritmo y se especifica su expresión en notación O:

1. Preparar conjunto de datos de entrada:  $O(n)$ 
  - Dependerá del costo de preprocesamiento de las imágenes y de la división en conjuntos de entrenamiento, validación y prueba.
  - Si el preprocesamiento de imágenes es lineal con respecto al número de píxeles y la división del conjunto es también lineal, la complejidad total podría considerarse lineal, expresándose como  $O(n)$ , donde  $n$  es el número de píxeles o tamaño de la imagen.
2. Aplicar el método de explicabilidad a las imágenes:  $O(n) + O(m * n) + O(r * k)$

- Estará dada principalmente del tamaño de la red y el costo de pasar las imágenes a través de esta. Dado que la red explicadora es relativamente pequeña, la complejidad de pasar las imágenes a través de esta podría considerarse lineal con respecto al número de píxeles. Por tanto sería una complejidad  $O(n)$ .

- Sin embargo, si la red pre-entrenada es compleja (por ejemplo, VGG19) y/o las imágenes son grandes, la complejidad será significativa. En este caso, se puede expresar la complejidad como  $O(m * n)$ , donde  $m$  es la complejidad de la red pre-entrenada.

- Dado que los pesos del modelo están congelados y no se realiza un entrenamiento completo, la complejidad debería ser relativamente eficiente.

- Además, es importante considerar el impacto de las operaciones repetitivas y cómo afectan a la complejidad total. Esto podría expresarse por ejemplo para el entrenamiento como  $O(r * k)$ , donde  $r$  es el número de repeticiones y  $k$  la complejidad asociada con las operaciones dentro del ciclo de entrenamiento.

Si a estos pasos se suma guardar los resultados, la complejidad aumenta aún más, ya que estará dada por un ciclo a lo largo del conjunto de datos y por cada posición de este, se evalúa la imagen a través de las capas del modelo y se guardan tres imágenes con su respectiva información. Esta complejidad puede expresarse como  $O(d * p)$ , donde  $d$  es el tamaño del conjunto de imágenes y  $p$  la complejidad asociada las operaciones dentro del ciclo de guardar.

Finalmente, al resumir la complejidad total a partir de las obtenidas anteriormente resulta:  $O(n) + O(n) + O(m * n) + O(r * k) + O(d * p)$ . Es posible simplificar esto sumando las complejidades lineales y agrupando términos similares a:  $O(n + m * n + r * k + d * p)$ . En esta expresión cada término está relacionado con:  $n$ , el tamaño de las imágenes.

$m * n$ , la complejidad de la red preentrenada.

$r * k$ , las operaciones repetitivas durante el entrenamiento.

$d * p$ , el ciclo de guardar resultados.

### 2.3. Alcance y limitaciones del método propuesto

El método propuesto se considera un método dependiente del modelo, ya que necesita adherirse a un modelo entrenado específico para aprender a partir de sus predicciones, por lo que aprende de imágenes con las transformaciones requeridas por este modelo en particular, de las etiquetas de su salida y sus pesos.

El enfoque de este puede caracterizarse por pertenecer al de mapas de saliencia y por proporcionar explicaciones locales, ya que se centra en la interpretación de la importancia de cada píxel buscando los rasgos que más influyeron en una predicción particular.

La implementación del método, explicada anteriormente tiene particularidades que se pueden variar en futuras pruebas, pero en este caso se utiliza todo acorde al modelo pre-entrenado VGG19 y al subconjunto de datos del conjunto Food-101. De igual forma, otros detalles como algunos hiperparámetros es posible que se encuentren combinaciones que mejoren los resultados tras una búsqueda exhaustiva de estos, proceso conocido como grid search.

Dado que la investigación se encuentra en la etapa de obtención de los primeros resultados experimentales, a partir de estos y de todo el trayecto del desarrollo, pueden resumirse algunas limitaciones que posteriormente se deben estudiar en búsqueda de la posibilidad de erradicarlas:

- Sensibilidad al conjunto de datos: dado que aprender a perturbar imágenes específicas del conjunto de datos de entrenamiento, podría volver al modelo sensible a las características y patrones únicos de ese conjunto. Esto podría limitar su generalización a otros conjuntos de datos.
- Se requieren recursos computacionales significativos para el entrenamiento de forma óptima y la inferencia.

## 2.4. Diferencias con respecto a los enfoques actuales

Los enfoques actuales comunes fueron mencionados en el capítulo anterior, cada método en dependencia de su tipo tiene formas específicas de proceder, sin embargo, podría generalizarse en que se basan fundamentalmente en el cálculo a través del uso de funciones matemáticas, teorías de juegos, aproximaciones, variaciones de parámetros.

En contraste con estos enfoques actuales, el método propuesto trae a la vida una idea diferente y es el uso de redes neuronales para explicar las decisiones de la red entrenada para clasificar imágenes. Ciertamente la técnica de perturbación de los píxeles para determinar como afecta este proceso a la salida no es nuevo, al igual que los enfoques utilizados en la definición de la red, pero sí lo constituye el hecho de enseñar a una red a perturbar ella misma la imagen basándose en los pesos de sus píxeles.

## 2.5. Conclusiones parciales

Se desarrolló el nuevo método nombrado como VZN (del inglés Vision's eXplainer Network), local y específico del modelo, para interpretar los resultados de la clasificación de imágenes utilizando las facilidades que brinda Pytorch para todo lo relacionado a redes neuronales. Además se utilizaron otras bibliotecas auxiliares de Python como Matplotlib, Imageio, útiles para la visualización y el manejo de las imágenes respectivamente.

Se implementaron las redes y funciones necesarias para el preprocesamiento, entrenamiento, entre otros procesos que se realizan en este caso adaptados al modelo VGG19 entrenado en el conjunto de datos escogido. Sin embargo, es fácil de adaptar a otros casos, implementando un código semejante, cambiando adecuadamente la descarga del conjunto de datos o el modelo pre-entrenado.

VZN constituye un nuevo enfoque de explicación, sensible a los datos y tiene un costo computacional alto.

# **Capítulo 3**

# **Capítulo 3**

## **Evaluación del método propuesto en los casos de estudio**

En este capítulo se describen detalladamente los resultados de evaluar el método a partir de los tres casos de estudios definidos, desglosando sus especificaciones y revelando los matices de la toma de decisiones en escenarios diversos. La esencia de la investigación se manifiesta en la capacidad de comprender y explicar el proceso subyacente que guía la toma de decisiones del modelo entrenado VGG19. A través de la visualización detallada de este proceso, se busca arrojar luz sobre la opacidad inherente de los modelos de clasificación de imágenes como el escogido.

### **3.1. Descripción del conjunto de datos de entrada**

Se seleccionó el conjunto de datos “Food101” disponible en la biblioteca torch.datasets, conocido por su diversidad y complejidad en la clasificación de alimentos. Este se compone de imágenes de alimentos recopiladas de la web y se ha utilizado ampliamente en la investigación de reconocimiento de alimentos. Consta de 101 clases distintas, cada una de ellas cuenta con 750 imágenes para entrenamiento y 250 imágenes para pruebas, todas revisadas manualmente, conformando un total de 101 000 imágenes variadas. desde platos gourmet hasta aperitivos cotidianos, presentando desafíos significativos para los modelos de clasificación.

La resolución de las imágenes en este conjunto es de 512x512 píxeles y su organización sigue una estructura de carpetas, donde cada carpeta representa una clase específica de alimento, esta estructura facilita la carga y el procesamiento del conjunto de datos mediante herramientas como torchvision.datasets. Ver documentación para obtener detalles precisos y actualizados disponible en <https://pytorch.org/vision/stable/datasets.html#food101>.

Para adaptar el método de explicabilidad a escenarios más específicos, optamos por seleccionar subconjuntos particulares de Food101 en tres casos de estudio distintos. Esta estrategia permite explorar la capacidad del modelo para proporcionar explicaciones claras y comprensibles en contextos más restringidos. Este proceso no solo simplifica la tarea de clasificación, sino que también introduce variaciones de complejidad para

evaluar la robustez y adaptabilidad del método.

## 3.2. Transformaciones

Previo a la formación de cada conjunto de datos de los casos de estudio, se aplican transformaciones fundamentales para preparar las imágenes y se realizan ajustes específicos en la configuración de la capa de salida de VGG19. Este proceso de preparación es esencial para garantizar resultados fiables y coherentes durante la evaluación y explicabilidad del modelo. Antes de la formación de cada conjunto de datos, las imágenes son sometidas a un proceso de transformación uniforme. La transformación incluye:

- Redimensionamiento a 256x256 píxeles: Para establecer un tamaño base que permitiera una futura centralización y recorte.
- Recorte al centro, a 224x224 píxeles: Establecimiento del tamaño final que VGG19 acepta como entrada estándar.
- Conversión a Tensores: Las imágenes fueron convertidas en tensores para facilitar su procesamiento por modelos de aprendizaje profundo.

Estas transformaciones aseguran que todas las imágenes estén normalizadas y ajustadas al formato requerido por VGG19, preparándolas para la evaluación y el proceso de explicabilidad.

Para cada caso de estudio, el conjunto de datos se divide en tres partes: entrenamiento, validación y prueba. Además, se llevan a cabo las siguientes configuraciones específicas:

1. Se aplica la transformación definida a todas las imágenes, esta se realiza solamente para el primer caso de estudio, ya que el resto utiliza un subconjunto de estas ya transformadas.
2. Se configura VGG19 para clasificar el número determinado de clases.
3. Se entrena el modelo para guardarla y poder cargarlo posteriormente para los diferentes números de clases definidos.
4. Se evalúan las imágenes y se crean nuevas carpetas excluyendo las mal clasificadas. Estas imágenes filtradas se guardan y se utilizan como base para los casos de estudio del método propuesto.

Esta estrategia garantiza la generalización y reutilización de las imágenes filtradas para el entrenamiento del modelo orientado a la explicabilidad de la clasificación, evitando sesgo y optimizando el proceso de evaluación y explicación del modelo.

En general, para el entrenamiento del modelo propuesto con los datos mencionados anteriormente se hicieron pruebas experimentales utilizando dos variantes de

optimizadores *SGB*<sup>1</sup> y *Adam*<sup>2</sup>, obteniendo mejores resultados con la primera opción. Para ver mejor la definición de optimizadores en pytorch es posible consultar la documentación oficial disponible en <https://pytorch.org/docs/stable/optim.html>.

Por otro lado se define como función de pérdida *CrossEntropyLoss()*, utilizada anteriormente en el modelo pre-entrenado, ya que es una función empleada comúnmente en problemas de clasificación con múltiples clases, calcula la pérdida de entropía cruzada entre las salidas y las etiquetas de entrada, para más información consultar documentación disponible en <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>.

### 3.3. Caso de estudio 1

La selección de clases para este caso de estudio se realiza a partir de las 10 clases mejor clasificadas en términos de rendimiento del modelo entrenado. Esta selección inicial proporciona una visión detallada de cómo se aborda la clasificación en un subconjunto de clases de mejor rendimiento, puede ser que estas tengan características distintivas, patrones visuales únicos o simplemente sean más fácilmente distinguibles para VGG19 en comparación con otras clases, ofreciendo un punto de partida sólido para la explicabilidad del modelo.

- Tamaño del conjunto de entrenamiento: 5912
- Tamaño del conjunto de validación: 1976
- Tamaño del conjunto de prueba: 1958

Para definir cada *DataLoader*<sup>3</sup> de cada conjunto se utilizan los parámetros:

- *batch\_size*<sup>4</sup>: 32
- *num\_workers*<sup>5</sup>: 2

Para el entrenamiento el valor del número de épocas se varía en un rango de 4 a 16, observándose que los valores más altos no mejoraban significativamente los resultados, en algunos casos se lograba lo contrario, y aumentaba considerablemente el tiempo de ejecución, por lo cual se determina apropiado utilizar 5. Durante el proceso de

<sup>1</sup>SGD (Stochastic Gradient Descent): Este optimizador actualiza los parámetros del modelo en la dirección opuesta al gradiente de la función de pérdida con respecto a los parámetros.

<sup>2</sup>Adam: Este optimizador combina las ventajas del método de SGB con momentum y el método de RMSProp. Adam adapta la tasa de aprendizaje para cada parámetro de forma adaptativa, lo que puede llevar a una convergencia más rápida y a una mejor generalización.

<sup>3</sup>Un *DataLoader* en PyTorch es una clase que representa un iterable de Python sobre un conjunto de datos. Este combina un conjunto y un muestreador, y proporciona un iterable que proporciona acceso fácil a las muestras, ya que se puede configurar para cargar datos en lotes y reordenar el conjunto de datos

<sup>4</sup>*batch\_size* se refiere al número de muestras que se utilizan en cada iteración durante el proceso determinado

<sup>5</sup>*num\_workers* determina cuántos subprocesos se utilizarán para cargar los datos en paralelo

aprendizaje, se guarda el valor de la pérdida promedio(Loss) por época(epoch), para observar el progreso en este caso de estudio. Ver gráfico 3.1

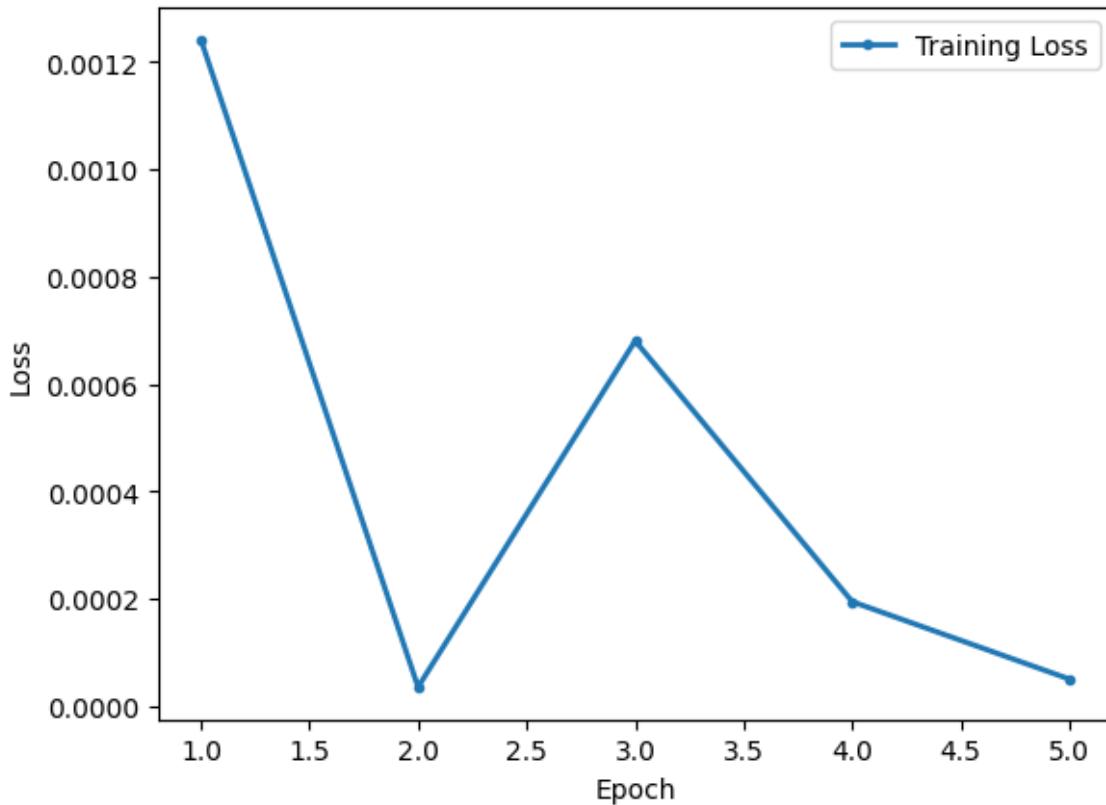


Figura 3.1: Gráfico de la pérdida promedio por época durante el entrenamiento del caso de estudio 1

Para evaluar el modelo recién entrenado y paralelamente guardar los datos se utiliza la función **save** definida. A partir de esto, se puede acceder entonces a cualquier imagen del conjunto de datos evaluado para poder saber su predicción, mapa de relevancia y la imagen perturbada. Luego de esto, es de suma importancia la función **show\_images**, para visualizar los resultados mostrando ejemplos a partir de los resultados guardados. A continuación en la figura 3.2 y 3.3 se muestran dos de ellos.

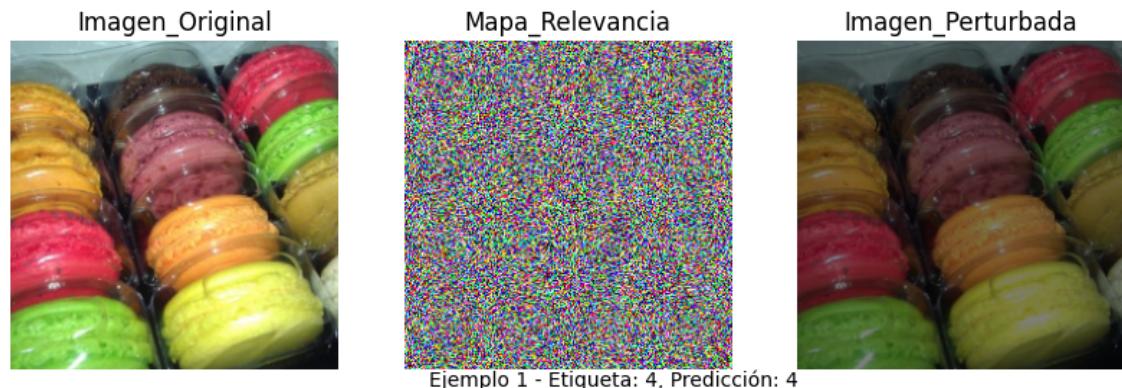


Figura 3.2: Visualización del ejemplo 1

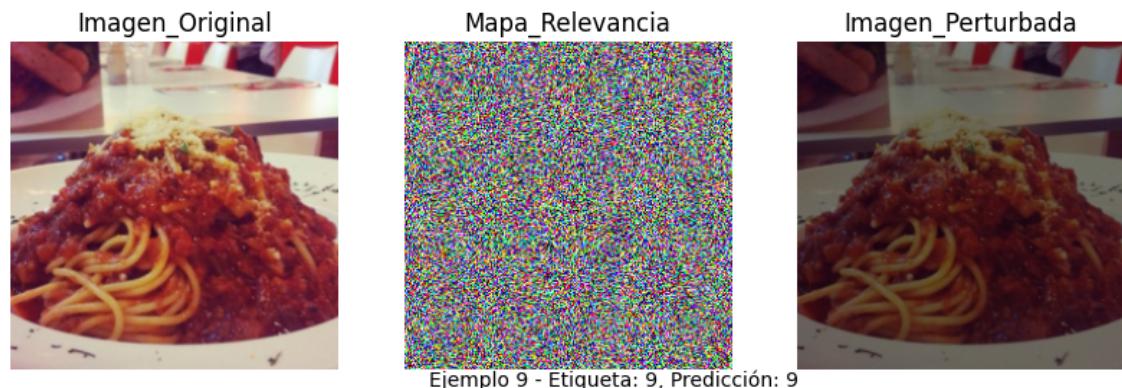


Figura 3.3: Visualización del ejemplo 2

### 3.4. Caso de estudio 2

Para este caso de estudio se reduce aún más el conjunto de imágenes seleccionando aleatoriamente un subconjunto de 5 clases de alimentos a partir de las 10 mejor clasificadas del caso anterior. Este enfoque busca entrenar y evaluar la capacidad del modelo para proporcionar explicaciones en contextos más limitados con respecto a las evaluaciones realizadas.

- Tamaño del conjunto de entrenamiento: 2951
- Tamaño del conjunto de validación: 986
- Tamaño del conjunto de prueba: 942

Para definir cada DataLoader en este caso se utilizan los parámetros:

- batch\_size: 32
- num\_workers: 2

Para el entrenamiento en este caso de estudio se utilizan 8 épocas y la pérdida promedio por época se puede ver en la figura 3.4

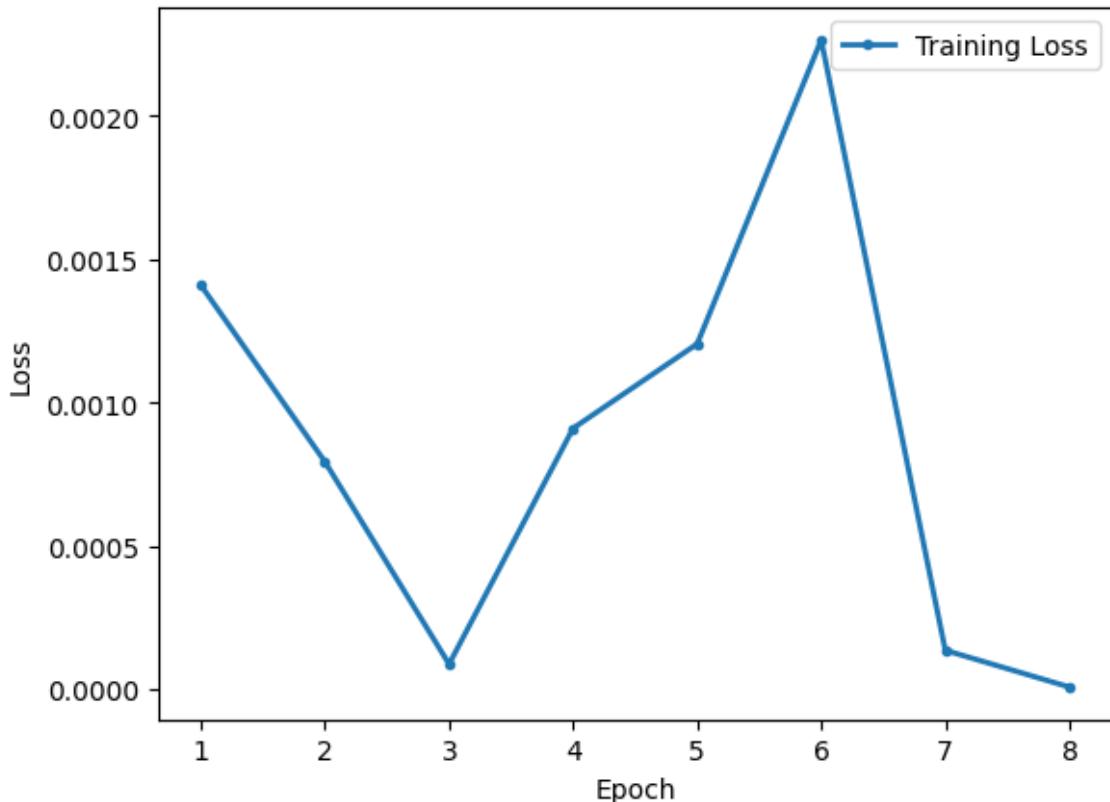


Figura 3.4: Gráfico de la pérdida promedio por época durante el entrenamiento del caso de estudio 2

Posteriormente, al guardar los resultados y visualizar se obtiene información como la que se muestra a continuación en el siguiente ejemplo.

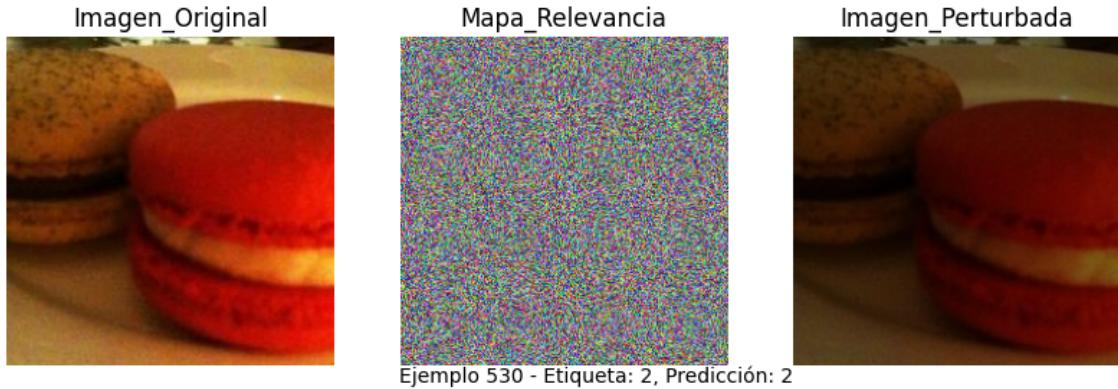


Figura 3.5: Visualización del ejemplo 1

### 3.5. Caso de estudio 3

En el último caso de estudio, nuevamente se selecciona aleatoriamente, esta vez un subconjunto de solo 2 clases de alimentos de las 10 mejor clasificadas. Esta configuración específica brinda la oportunidad de evaluar el método en un escenario de clasificación de imágenes mínimo. Además este enfoque consume una cantidad menor de recursos, facilitando la realización de pruebas adicionales y ajustes de parámetros con tiempos de ejecución más rápidos.

- Tamaño del conjunto de entrenamiento: 755
- Tamaño del conjunto de validación: 766
- Tamaño del conjunto de prueba: 400

En este caso de estudio, manteniendo `num_workers` del `DataLoader` = 2, se hicieron variaciones en la definición de la función de perturbación, el tamaño del batch, en la cantidad de neuronas de salida de la segunda capa del modelo, y como anteriormente, en la cantidad de épocas, haciendo posible visualizar a continuación algunos de estos resultados para analizar la existencia o no de cambios significativos entre ellos. Se muestra en un primer lugar la línea de progreso del entrenamiento a través del análisis de la pérdida promedio por cada época, se especifican en la parte inferior las variaciones realizadas para el caso específico y seguido de esto, la imagen a través de las diferentes etapas del método: entrada, confección del mapa de relevancia y perturbación. Ver figuras 3.6, 3.7, 3.8, 3.9 y 3.10.

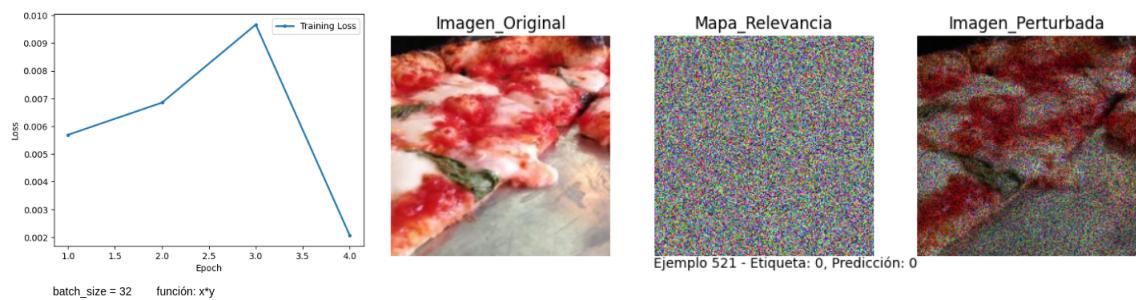


Figura 3.6: Ejemplo 1

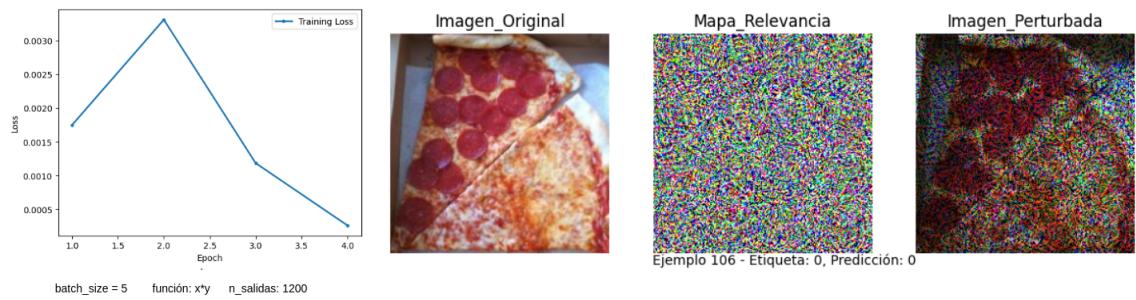


Figura 3.7: Ejemplo 2

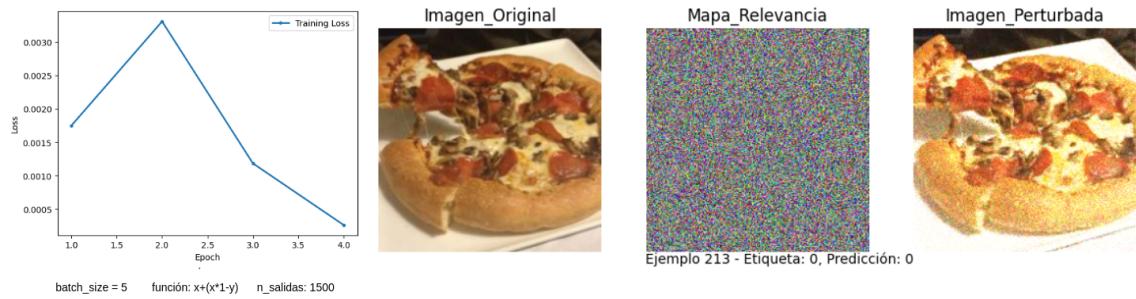


Figura 3.8: Ejemplo 3

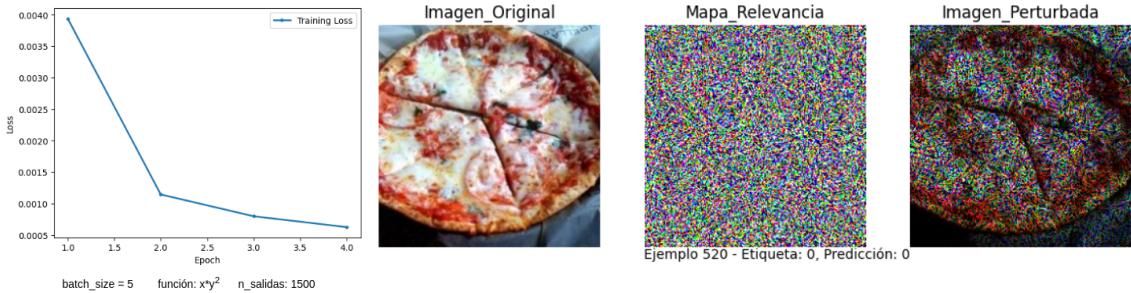


Figura 3.9: Ejemplo 4

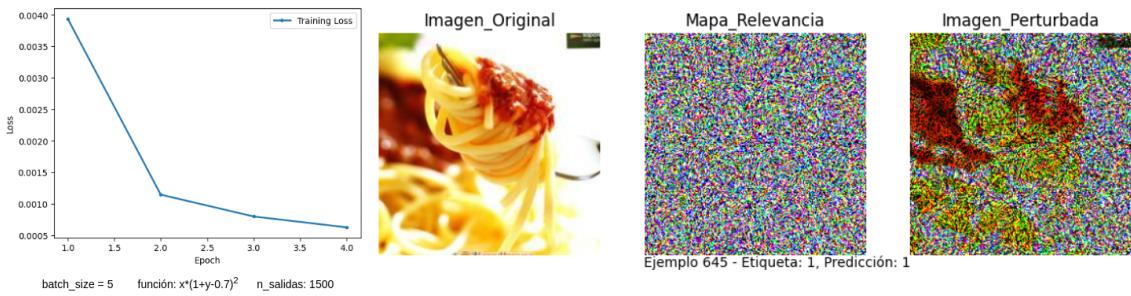


Figura 3.10: Ejemplo 5

La cantidad de épocas a lo largo de los experimentos en diferentes casos, no ha mostrado mejoras relevantes al aumentarla en el rango posible, por lo cual por una cuestión de recursos se ajusta a 4. Cada ejemplo que se muestra corresponde a los resultados a partir de los cambios especificados.

El hecho de que no exista una marcada diferencia entre los resultados, traza la posibilidad que todos sean válidos y se deba profundizar en futuras investigaciones para restringir los escenarios de prueba lo mejor posible.

### 3.6. Comparación de los casos de estudio

Cada caso de estudio se diseñó de manera única, introduciendo variaciones en términos de duración de entrenamiento, tamaño de lote (batch size), funciones de perturbación y configuración de la red neuronal. Todos los resultados mostraron una interpretación de los mapas de relevancia con cierta falta de detalle, pero que marcan un punto de partida en el desarrollo de la investigación del método propuesto, ya que se encuentra en su etapa experimental. La diferencia entre las visualizaciones de los casos 1 y 2 no fue significativamente notable y la imagen perturbada se aprecia como la original pero con un tono más oscuro, lo que da a entender que se perturban algunos píxeles aislados bajando su intensidad en la mayoría. En el tercer caso resultaron imágenes perturbadas

más detalladas y unos colores más acentuados en los mapas de relevancia. Esto inclina la investigación a que es posible obtener mejores resultados con la reducción del conjunto de entrada a una menor cantidad de clases a predecir, un aumento del número de neuronas en la segunda capa lineal del modelo y al utilizar un tamaño de lote pequeño. Aunque los resultados no mostraron una diferencia marcada entre los casos de estudio, es importante destacar que todos los escenarios pueden considerarse válidos. La falta de divergencia significativa podría sugerir la robustez del método ante diversas configuraciones. No obstante, se reconoce la necesidad de futuras investigaciones para refinar los escenarios de prueba y profundizar en la comprensión de las interacciones entre los parámetros del modelo, cada función de perturbación y la interpretación de la explicabilidad.

### **3.7. Conclusiones parciales**

Se exploraron los resultados obtenidos al evaluar el método a través de tres casos de estudio distintos y de la visualización detallada de este proceso, lo cual ha buscado arrojar luz sobre la opacidad inherente de los modelos de clasificación.

Las transformaciones aplicadas a las imágenes, junto con las configuraciones específicas para cada caso de estudio, fueron esenciales para preparar los datos y garantizar resultados fiables y coherentes durante la evaluación y explicabilidad del modelo.

Se observó cierta consistencia en la interpretación de los mapas de relevancia, es decir, aunque con falta de detalle en la mayoría de los casos, se puede caracterizar como un primer avance ya que el modelo aprende y confecciona un mapa de relevancia que no es nulo. La diferencia más notoria se encontró en el tercer caso de estudio, donde las imágenes perturbadas mostraron un nivel de detalle superior y colores más acentuados en los mapas de relevancia.

La comparación entre los casos de estudio reveló que, la reducción del conjunto de entrada a un menor número de clases y el ajuste de ciertos parámetros del modelo parecen tener un impacto positivo en la calidad de las explicaciones proporcionadas.

# **Conclusiones**

# Conclusiones

En el ámbito de la Inteligencia Artificial, la comprensión de los modelos de caja negra es esencial para garantizar transparencia y confiabilidad en su desempeño. Tradicionalmente, los métodos de evaluación de la calidad de estos modelos han sido limitados, siendo insuficientes para revelar plenamente su funcionamiento interno. Reconociendo esta carencia, se propuso un enfoque innovador que busca mejorar la interpretabilidad y comprensión de estos modelos para el caso de la clasificación de imágenes.

- Se desarrolló un nuevo método local y dependiente del modelo para interpretar la clasificación de imágenes: VZN. Este constituye un nuevo enfoque de explicación basado en redes neuronales, sensible a los datos y con un alto costo computacional
- Se implementó una red en cascada que comunica el método propuesto con el modelo entrenado VGG19 de forma que el aprendizaje está orientado a los píxeles que fueron relevantes para la predicción hecha por el modelo. La implementación desarrollada incluyó funciones necesarias para el preprocesamiento, entrenamiento, entre otros procesos que se realizan en este caso adaptados al modelo entrenado con las facilidades de Pytorch, pero fácil de adaptar a otros casos
- Como resultado de la evaluación del método se obtienen resultados adecuados siendo mejores donde se utilizó el conjunto de datos perteneciente a dos clases y se determinó que el ajuste de ciertos parámetros del modelo como el número de neuronas de salida y el tamaño del batch parecen tener un impacto positivo en la calidad de las explicaciones proporcionadas.

# Recomendaciones

- Realizar una búsqueda de hiperparámetros para mejorar el entrenamiento de la red explicadora.
- A partir de la búsqueda hiperparámetros, evaluar los efectos de las diferentes funciones definidas en los casos de estudio conformados:  $x * y$ ,  $x + (x * 1 - y)$ ,  $x * y^2$ ,  $x * (1 + y - 0.7)^2$ .
- Evaluar los efectos de aplicar una transformación a las imágenes que les realce el brillo y el contraste.
- Aplicar el método de explicación para otros modelos de clasificación de imágenes.

## Referencias

- Aha, D. W. and Kibler, D. (1991). *Lazy Learning*. Kluwer Academic Publishers.
- Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., Samek, W., Müller, K.-R., Dähne, S., and Kindermans, P.-J. (2019). innvestigate neural networks! *Journal of Machine Learning Research*, 20(93):1–8.
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., and Al-Amidie, M. (2021). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1):53.
- Amador, R. (2023). Extensión de la biblioteca innvestigate con métodos para evaluar la calidad de sus explicaciones. *Universidad Central “Marta Abreu” de Las Villas, Cuba*.
- Amesoeder, C., Hartig, F., and Pichler, M. (2023). cito: An r package for training neural networks using torch.
- Anwar, A. (2019). Difference between alexnet, vggnet, resnet, and inception. *Towards Data Science*. Disponible en: <https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecc96>.
- Apley, D. W. and Zhu, J. (2019). Visualizing the effects of predictor variables in black box supervised learning models.
- Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q. V., Luss, R., Mojsilović, A., et al. (2019). One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*.
- Badalia, S. (2021). Model interpretability: Demystifying black-box models. Acuity Knowledge Partners. Available at <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9343258/>.
- Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press.
- Basogain Olabe, X. (2003). *Redes Neuronales Artificiales y sus aplicaciones*. Servicio Editorial de la Universidad del País Vasco.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Carvalho, D. V., Marques Pereira, E., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*.
- Chollet, F. (2021). *Deep Learning with Python*. Manning Publications, Shelter Island, NY, 2nd edition.

- Cotino Hueso, L. and Castellanos Claramunt, J. (2023). *Transparencia y explicabilidad de la inteligencia artificial*. Editorial Tirant, Valencia.
- de Boves Harrington, P., Wan, C., and Clippinger (2002). Applied to artificial neural networks : What has my neural network actually learned.
- Dietterich, T. G., Maass, W., Simon, H. U., and Warmuth, M. K. (2008). Theory and praxis of machine learning. *Semantic Scholar*.
- Díaz-Ramírez, J. (2021). Aprendizaje automático y aprendizaje profundo. *Ingeniare. Revista chilena de ingeniería*, pages 180–181.
- Gandikota, R. (2019). Understanding convolutional neural networks. *Machine Learning*.
- Gong, Z., Zhong, P., Yu, Y., Hu, W., and Li, S. (2019). A cnn with multiscale convolution and diversified metric for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 57:3599–3618.
- Grieve, P. (2023). Deep learning vs. machine learning: What's the difference? *Zendesk Blog*.
- Haykin, S. (2009). *Neural Networks and Learning Machines*. Pearson Education.
- Kilpi, E. (2017). *Neural networks as the architecture of human work*. Springer.
- Kriegeskorte, N. and Golan, T. (2019). Neural network models and deep learning. *Current Biology*, 29(7):R231–R236.
- Lanjewar, M. G. and Gurav, O. L. (2022). Convolutional neural networks based classifications of soil images. *Multimedia Tools and Applications*, 81:10313 – 10336.
- Le, J. (2018). The 4 convolutional neural network models that can classify your fashion images. *Towards Data Science*. Disponible en: <https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images>
- Lusim, L. and Marchesano, P. (2023). Inteligencia artificial explicable: tecnología y transparencia para la industria 4.0. *Universidad ORT Uruguay*.
- Manjarrez, L. (2014). Relaciones neuronales para determinar la atenuación del valor de la aceleración máxima en superficie de sitios en roca para zonas de subducción. Master's thesis, The University of Arizona.
- Modi, P. (2023). Convolutional neural networks for dummies. <https://towardsai.net/p/deep-learning/convolutional-neural-networks-for-dummies>.
- Molnar, C. (2019). *Interpretable Machine Learning*. Leanpub, Canada.
- Molnar, C. (2022). *Interpretable Machine Learning*. Leanpub, 2 edition.
- Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *digital signal processing*. Elsevier Inc., 73.

- Moreno, A. N. (2020). Introducción a las Redes Neuronales aplicadas al Aprendizaje Supervisado. *Hackers and Developers™ Press*.
- Nguyen, T.-H., Nguyen, T.-N., and Ngo, B.-V. (2022). A vgg-19 model with transfer learning and image segmentation for classification of tomato leaf disease. *AgriEngineering*, 4(4):871–887.
- Ongsulee, P. (2017). Artificial intelligence, machine learning and deep learning. *IEEE*.
- Onose, E. (2023). Explainability and auditability in ml: Definitions, techniques, and tools. *Electronics*. Disponible en: <https://neptune.ai/blog/explainability-auditability-ml-definitions-techniques-tools>.
- O’Sullivan, C. (2022). What are model agnostic methods? *Towards Data Science*.
- Ramírez Véliz, R. B., L. S. S. C. . G. B. J. M. (2022). El humano y la máquina: perspectivas sobre inteligencia artificial, agentes y sistemas inteligentes. *RECIAMUC*, 6(3).
- Russell, S. and Norving, P. (2020). *Artificial Intelligence: A Modern Approach*. Prentice Hall.
- Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K.-R. (2021). Explaining deep neural networks and beyond: A review of methods and applications. *IEEE*.
- Samek, W., Wiegand, T., and Muller, K.-R. (2019). *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*. Springer.
- Sarker, I. H. (2021). *Machine Learning: Algorithms, Real-World Applications and Research Directions*. Springer, New York, NY.
- Sekhar, A., B. S. H. R. S. A. K. M. A. and Yang, L. (2022). Brain tumor classification using fine-tuned googlenet features and machine learning algorithms. *IEEE journal of biomedical and health informatics*, 26(3):983 – 991.
- Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning.
- Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Pearson Education.
- Taye, M. M. (2023). Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions. *Computers*, 12(5).
- Wang, Y., Wang, Y., and Chen, Y. (2021). Eeg functional connectivity analysis based on neural network and granger causality. *Journal of Ambient Intelligence and Humanized Computing*, 12(7):6935–6946.

- Wang, Y. and Zhang, H. (2023). Identification of winter wheat pests and diseases based on improved convolutional neural network. *BMC Bioinformatics*.
- Xu, Y., Liu, X., Cao, X., Huang, C., Liu, E., Qian, S., Liu, X., Wu, Y., Dong, F., Qiu, C.-W., Qiu, J., Hua, K., Su, W., Wu, J., Xu, H., Han, Y., Fu, C., Yin, Z., Liu, M., Roepman, R., Dietmann, S., Virta, M., Kengara, F., Zhang, Z., Zhang, L., Zhao, T., Dai, J., Yang, J., Lan, L., Luo, M., Liu, Z., An, T., Zhang, B., He, X., Cong, S., Liu, X., Zhang, W., Lewis, J. P., Tiedje, J. M., Wang, Q., An, Z., Wang, F., Zhang, L., Huang, T., Lu, C., Cai, Z., Wang, F., and Zhang, J. (2021). Artificial intelligence: A powerful paradigm for scientific research. *The Innovation*, 2(4):100179.
- Zhang, D. J., Li, K., Chen, Y., Wang, Y., Chandra, S., Qiao, Y., Liu, L., and Shou, M. Z. (2021). Morphmlp: A self-attention free, mlp-like backbone for image and video. *ArXiv*, abs/2111.12527.
- Zhang, X., Chen, M., Zhang, Z., and Lu, S. (2022). A texture detail-oriented generative adversarial network: motion deblurring for multi-textured images. *Applied Intelligence*, 53.
- Zhang, Y. and Zhang, J. (2019). A neural network-based phishing website detection method. *IEEE Access*, 7:102653–102663.
- Zhou, J., Gandomi, A. H., Chen, F., and Holzinger, A. (2021). Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5):593.
- Zhou, Q. and Purvis, M. K. (2004). A market-based rule learning system. In *Australasian Computer Science Week*.
- Zielinski, P., Krishnan, S., and Chatterjee, S. (2020). Weak and strong gradient directions: Explaining memorization, generalization, and hardness of examples at scale. *arXiv: Learning*.