



UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Trabajo Fin de Máster

Redes neuronales adversarias en seguridad informática

Alumno: Antonio Mudarra Machuca

Tutor: Antonio Jesús Rivera Rivas

María José del Jesus Díaz

Dpto.: Departamento de informática

septiembre, 2023



UNIVERSIDAD DE JAÉN

D. Antonio Jesús Rivera Rivas y D^a María José del Jesus Díaz, tutores del Trabajo Fin de Grado titulado: **Redes neuronales adversarias en seguridad informática**, que presenta Antonio Mudarra Machuca, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2023

Estudiante

Tutores

Antonio Mudarra
Machuca

Antonio Jesús Rivera
Rivas

María José del Jesus Díaz

Agradecimientos

Incluir aquí los agradecimientos que se deseen, dedicatoria y cualquier otro texto de carácter personal.

Tabla de contenidos

Notación	2
Teoremas	6
1. INTRODUCCIÓN	7
1.1. Introducción	7
1.2. Motivación	8
1.3. Objetivos primarios	9
1.4. Objetivos secundarios	9
1.5. Metodología y planificación del proyecto	9
1.5.1. Metodología	10
1.5.2. Planificación	10
1.5.3. Presupuesto	10
1.6. Estructura de la memoria	10
2. ANTECEDENTES Y ESTADO DEL ARTE	11
2.1. Introducción	11
2.2. Antecedentes	12
2.2.1. Historia, línea temporal	12
2.2.2. Historia de la Inteligencia Artificial	14

2.3. Estado del arte	19
2.3.1. La ciencia de datos	19
2.3.2. La minería de datos	20
2.3.3. Aprendizaje automático - <i>Machine Learning</i>	22
2.3.4. Aprendizaje profundo - <i>Deep Learning</i>	24
2.3.5. Redes neuronales artificiales - <i>Artificial Neuronal Networks</i> . . .	24
2.3.5.1. Elementos de una neurona artificial	24
2.3.5.2. Funcionamiento de una red neuronal	25
2.3.5.3. El descenso de gradiente	28
2.3.5.4. Optimizadores	31
2.3.5.5. Funciones de activación	31
2.3.5.6. Tipos de capas	34
2.3.5.7. Topologías de redes neuronales	36
2.3.6. Redes generativas adversariales - <i>Generative Adversarial Networks (GAN)</i>	37
2.3.7. La seguridad de las redes neuronales	40
2.3.8. Amenazas de <i>deep learning</i>	42
2.3.8.1. Evasión	42
2.3.8.2. Envenenamiento	43
2.3.8.2.1. Label Flipping Attacks	43
2.3.8.2.2. Clean Label Data Poisoning Attack	43
2.3.8.2.3. Backdoor Attack	43
2.3.8.3. Extracción	43
2.3.8.4. Inferencia	44

2.3.8.4.1. Inferencia por Atributos	44
2.3.8.4.2. Inferencia por Pertenencia	44
2.3.8.4.3. Inversión de modelos	44
2.3.8.4.4. Reconstrucción	44
2.3.9. Defensas contra los ataques en <i>deep learning</i>	44
2.3.9.1. Preprocesamiento	44
2.3.9.2. Post procesamiento	44
2.3.9.3. Entrenamiento	44
2.3.9.4. Transformación	44
2.3.9.4.1. Evasión	44
2.3.9.4.2. Envenenamiento	44
2.3.9.5. Detector	44
2.3.9.5.1. Evasión	44
2.3.9.5.2. Envenenamiento	44
2.3.10. Métricas en los ataques adversariales	44
2.3.11. Redes neuronales en <i>Red team</i> y <i>Blue team</i>	44
2.3.12. Las redes neuronales para la seguridad informática	46
2.3.13. Normativa y estándares	46

Notación

Notación	Definición
\mathbb{Z}	Números enteros. i.e. $\mathbb{Z} = \{\dots, -2, -1, 0, +1, +2, \dots\}$
\mathbb{Z}^+	Números enteros positivos. i.e. $\mathbb{Z}^+ = \{0, +1, +2, \dots\}$
\mathbb{Z}^{++}	Números enteros positivos sin el cero. i.e. $\mathbb{Z}^{++} = \{+1, +2, \dots\}$
\mathbb{N}	Números naturales
\mathbb{R}	Números reales
\mathbb{C}	Números complejos
R^n	Espacio vectorial n -dimensional de números reales
ϵ	Para cantidades, arbitrariamente pequeñas

Tabla. 1: Notación números y arrays

Notación	Definición
$\mathbf{a} = [a_i]_{i=1, \dots, n}$	Vectores definidos en minúscula, negrita y cursiva
$\mathbf{A} = [a_{i,j}]_{i=1, \dots, n, j=1, \dots, m}$	Matrices definidas en mayúscula, negrita y cursiva
\mathbf{A}	Tensores definidos en mayúscula, negrita y en estilo sans serif
\mathbf{a}_i	Elemento i del vector \mathbf{a} , empezando el índice por 1
$\mathbf{A}_{i,j}$	Elemento (i, j) de la matriz \mathbf{A}
$\mathbf{A}_{i,:}$	Fila i de la matriz \mathbf{A}
$\mathbf{A}_{:,j}$	Columna j de la matriz \mathbf{A}
$\mathbf{A}_{i,j,k}$	Elemento (i, j, k) 3D del tensor \mathbf{A}
$\mathbf{A}_{:,:,k}$	Desplazamiento 2D del Tensor 3D

Tabla. 2: Notación de índices para vectores, matrices y tensores

Notación	Definición
I_n	Matriz identidad de tamaño $n \times n$
$0_{n,m}$	Matriz de ceros de tamaño $n \times m$
$\underline{0}_n$	Vector de ceros de tamaño n
$1_{n,m}$	Matriz de unos de tamaño $n \times m$
$\underline{1}_n$	Vector de unos de tamaño n
e_i	Vector estandar o vector canonico. i.e. $v_x = (1, 0, 0), v_y = (0, 1, 0), v_z = (0, 0, 1)$

Tabla. 3: Notación de vectores, matrices y tensores especiales

Símbolo	Definición
\mathbf{A}^\dagger	Traspuesta de la matriz \mathbf{A}
$\text{rk}(\mathbf{A})$	Rango de la matriz \mathbf{A}
$\text{tr}(\mathbf{A})$	Traza de la matriz \mathbf{A}
$\det(\mathbf{A})$	Determinante de la matriz \mathbf{A}
$\text{Im}(\Phi)$	Imagen del mapeo lineal Φ
$\text{ker}(\Phi)$	Núcleo (espacio nulo) de un mapeo lineal Φ
$ \cdot $	Determinante o valor absoluto
$\ x\ _p$	Norma L^p de x
$\ x\ $	Norma L^2 de x
$\ \underline{x} - \underline{y}\ _q$	Distancia en la misma dimensión entre \underline{x} e \underline{y}
$\underline{x} \odot \underline{m}$	Operación por elementos de los vectores o matrices
$\langle \underline{z}, \underline{y} \rangle = \underline{z}' \underline{y} = \sum_{j=1}^N z_j y_j$	Producto escalar de vectores por columnas $\underline{z}, \underline{y} \in \mathbb{R}^N$

Tabla. 4: Notación de operaciones con vectores y matrices

Notación	Definición
$f : \mathbb{A} \rightarrow \mathbb{B}$ $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$	Una función f con dominio \mathbb{A} y rango \mathbb{B} Gradiente, es un vector que indica la dirección de mayor pendiente de una superficie en un punto dado
$\nabla f(\mathbf{a}) \in \mathbb{R}^n$ $\nabla \cdot f = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} + \frac{\partial f}{\partial z}$ $\nabla \times f = \left(\frac{\partial f}{\partial y} - \frac{\partial f}{\partial z}, \frac{\partial f}{\partial z} - \frac{\partial f}{\partial x}, \frac{\partial f}{\partial x} - \frac{\partial f}{\partial y} \right)$	Gradiente de la función $f : \mathbb{R}^n \rightarrow \mathbb{R}$ con entrada \mathbf{a} Divergencia Rotación
$f_* = \min_x f(x)$ $x_* \in \arg \min_x f(x)$	El valor de función más pequeño de f El valor x_* (conjunto de valores) que minimiza f

Tabla. 5: Notación de funciones

Notación	Definición
\mathcal{X}	Conjunto de datos, donde $\mathcal{X} \in \mathbb{R}^N$
N	La dimensión de la matriz de características es $\mathcal{X} = I \times K$
$\{x_1, x_2, \dots, x_n\}$	Dimensión del espacio muestral de entrada \mathcal{X}
$T \text{ o } \mathbb{X} = \{x^{(i)}, y^{(i)}\}$	Conjunto con n elementos
$V \text{ o } \mathbb{X}' = \{x'^{(i)}, y'^{(i)}\}$	Conjunto de datos de entrenamiento $D \subset \mathcal{X}$
I	Conjunto de datos de validación $H \subset \mathcal{X}$
K	Número de instancias, registros u observaciones
\mathcal{Y}	Número de atributos, características, entradas o predictores
η	Conjunto de clases del conjunto de datos \mathcal{X} , esto es, $K = \mathcal{Y} $. i.e. $\mathcal{Y} = \{1, 2, \dots, K\}$
\hat{y}	Tasa de aprendizaje
\hat{y}	Predicción del modelo
θ, ω	Parámetros del modelo
\mathcal{L}	Función de perdida
ϕ, φ	Función de activación
$f(\cdot)$	Modelo
$c(\cdot)$	Modelo clasificador
$p(\cdot)$	Modelo predictor

Tabla. 6: Notación de *Machine Learning*

Notación	Definición
x	Entrada original (limpia, sin modificar) de un dato
x'	Dato adversarial
y'	Clase objetivo de ejemplo adversario
δ	Perturbación

Tabla. 7: Notación de GAN

Notación	Definición
$Q(x)$	Función de densidad de probabilidad de la variable aleatoria x
$\mathbb{E}_{x \sim Q}$	Esperanza de $f(x)$ con respecto a la distribución de probabilidad de x según Q

Tabla. 8: Notación probabilidad y estadística

Teoremas

Teoremas 0.0.1 (Bayes). *Sean C , A dos eventos, y $P(C|A)$ la probabilidad de C dependiente de A . Entonces.*

$$P(C|A) = \frac{P(C|A)P(A)}{P(A)}$$

Teoremas 0.0.2 (Convergencia de perceptrón). *Si el conjunto de patrones de entrenamiento $\{x^1, z^1\}, \{x^2, z^2\}, \dots, \{x^n, z^n\}$ es linealmente separable entonces el Perceptrón simple encuentra una solución en un número finito de iteraciones, es decir, consigue que la salida de la red coincida con la salida deseada para cada uno de los patrones de entrenamiento.*

Capítulo 1

INTRODUCCIÓN

1.1. Introducción

El objetivo principal de este trabajo es el análisis en el campo de la inteligencia artificial [Artifical Intelligence \(AI\)](#) de los distintos tipos de ataques y defensas que se pueden aplicar a un proceso de aprendizaje automático [Machine Learning \(ML\)](#) de las redes neuronales, nos centraremos principalmente en el aprendizaje profundo [Deep Learning \(DL\)](#). Este trabajo tratará principalmente de clasificar y explorar la seguridad que cuentan los modelos actuales. Se han desarrollado una gran cantidad de modelos en los últimos años y existen una gran variedad de modelos muy distintos, aunque podemos clasificar los ataques de todos estos modelos en las siguientes categorías (evadir, envenenar, inferir o extraer).

Debemos comprender que la seguridad repercute en todo el proceso de creación y uso del modelo, desde la recogida de los datos, tratamiento, diseño, creación y ejecución del modelo. Un ataque puede estar dirigido al conjunto de datos con el que se entrenará el modelo, a la arquitectura de la red neuronal, a los pesos, etc. Además, un ataque puede dirigirse a descubrir muestras que produzcan resultados erróneos, por lo que los posibles vectores de ataque son muy variados y complejos.

La idea de hacer robustos los modelos es que las defensas detecten ataques a la vez que se mejora la solidez del aprendizaje, para no cometer fallos al introducir valores anómalos.

Cada uno de estos tipos de ataques tiene su nomenclatura y se debe definir correctamente, ya que de lo contrario puede ser muy ambiguo el tipo de ataque que se está realizando, el objetivo que busca el ataque y los métodos que están empleando.

Lo que buscaremos en este trabajo será definir, analizar y estructurar los distintos tipos de ataques y sus posibles defensas, con el objetivo final de detectar y defender los modelos para hacerlos más robustos y confiables para la ciudadanía. Además de la creación de una guía que pueda orientar a modelos más seguros y éticos.

1.2. Motivación

En la última década, se han logrado significativos avances en el campo de la inteligencia artificial. Sin embargo, a lo largo de este proceso, como suele ser habitual, se ha descuidado aspectos cruciales relacionados con la seguridad. Esto ha dado lugar a la creación de productos que implementan la inteligencia artificial, pero presentan vulnerabilidades, riesgos potenciales, como redes neuronales poco robustas, filtraciones de datos, incumplimiento normativo, modelos que presentaban respuestas ofensivas, discriminantes ante etnias, etc. Además de presentar poca o ninguna explicabilidad de los resultados que presentan.

Esto lleva a muchas amenazas y riesgos que pueden afectar a empresas u organizaciones que apliquen inteligencia artificial si no hacen una implementación adecuada.

Problemas de seguridad en inteligencia artificial.

- Alineación de la inteligencia artificial.
- Recopilación de datos.
- Bias y discriminación.
- Modelos poco robustos.
- Transparencia y explicabilidad.
- Escala de los modelos.
- Cumplimiento legal y normativo.
- Actualización continua.
- Detección de usos malintencionados.

Esto ha llevado a la creación de regulaciones de la inteligencia artificial que son muy vagas en sus conceptos de implementación. Una primera aproximación fue el libro blanco¹ de la inteligencia artificial en 2018 [1]

En este trabajo propondremos una guía similar a la matriz [MITRE](#) con buenas prá-

¹Se conoce como libros blancos a los documentos que publican los gobiernos en determinados casos para informar a los órganos legislativos o a la opinión pública con el objetivo de ayudar a los lectores a comprender un tema, resolver o afrontar un problema (por ejemplo diseñando una política gubernamental a largo plazo), o tomar una decisión. [Enlace](#).

ticas para la construcción de modelos más robustos y que cumplan con nuestros objetivos.

La necesidad de desarrollar un marco de inteligencia artificial fiable para todos los miembros de la unión europea llevo a la comisión europea a la creación de nuevas normativas con un enfoque basado en el riesgo. Por lo que exploraremos el análisis de seguridad y posibles soluciones.

1.3. Objetivos primarios

El objetivo principal de este trabajo es hacer un estudio bibliográfico del campo de los modelos adversarios aplicados a la seguridad informática, búsqueda de vectores de ataque en modelos, posibles formas de auditar los distintos modelos y costes de ataques por distintos modelos adversariales que tratamos en la sección “Estado del arte”.

1.4. Objetivos secundarios

Como objetivos secundarios se ha planteado el análisis y creación de un marco de trabajo para la implementación de modelos de inteligencia artificial fiable siguiendo los estándares normativos, legislativos y éticos que propone la unión europea, con el objetivo de facilitar la implementación de modelos fiables en los distintos estados miembros.

1.5. Metodología y planificación del proyecto

Una vez definidos los objetivos del tema de investigación y desarrollo, deberemos definir el alcance para estimar el tiempo requerido y el presupuesto para el proyecto. Recordemos que en la investigación puede ser del tipo exploratorio, buscando nuevos campos para alcanzar nuevos logros técnicos, confirmatoria para validar los resultados obtenidos en otras investigaciones o desarrollos, también puede ser una combinación de los tipos mencionados previamente. En nuestro caso se tratará de un proyecto híbrido, tanto de exploración como de confirmación.

1.5.1. Metodología

1.5.2. Planificación

Se ha de hacer una planificación estricta de las tareas, objetivos, hitos y dependencias, para ello usaremos los diagramas Gantt² con el objetivo de representar los hitos de esta investigación y del proyecto. Seguiremos una metodología *Scrum* y *Lean* fijando reuniones cada semana para tener un control sobre el avance de la investigación y el desarrollo de este proyecto.

Se ha definido en la Figura 1.1 la lista de tareas, fecha de inicio, duración, fecha de fin y el coste de trabajo usando la técnica de tallas de camisetas.

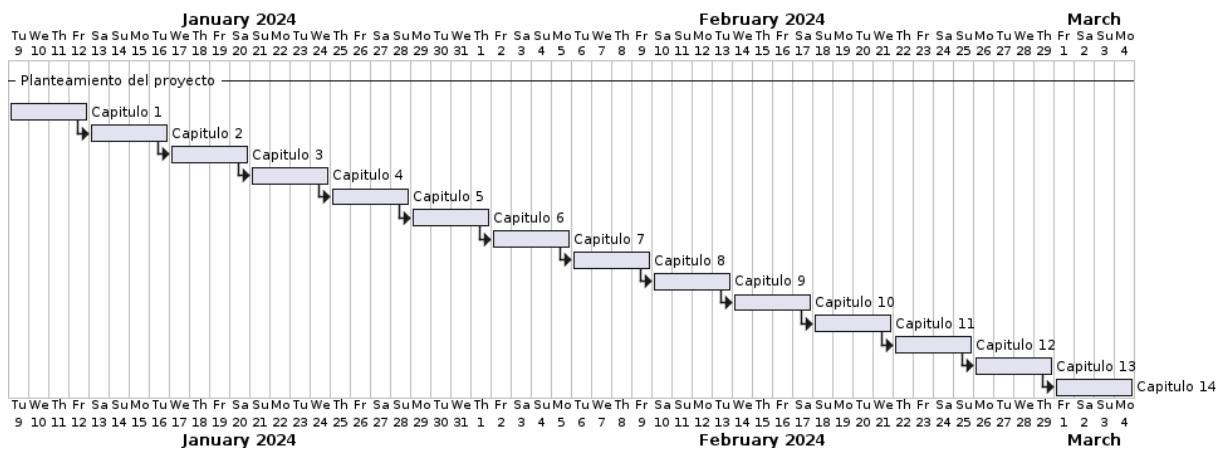


Figura 1.1: Diagrama Gantt

1.5.3. Presupuesto

1.6. Estructura de la memoria

En este trabajo, hemos utilizado la documentación proporcionada por **PyTorch** para explicar diversos conceptos de las redes neuronales. Nos referimos a la documentación oficial de PyTorch [2]. Usamos el término `torch`, `torch.nn` y `nn`.

²Representación gráfica de la evolución de un proyecto. [ASANA](#)

Capítulo 2

ANTECEDENTES Y ESTADO DEL ARTE

2.1. Introducción

El proyecto que desarrollamos se encuentra en una frontera muy difusa de múltiples ramas del conocimiento, siendo muy interdisciplinar, se trata de una revisión de los ataques, seguridad y robustez a las redes neuronales centrandonos en ataques adversariales. Por lo que debemos explicar que es la ciencia de datos, el proceso [Knowledge Discovery in Databases \(KDD\)](#), la inteligencia artificial generativa y la seguridad informática.

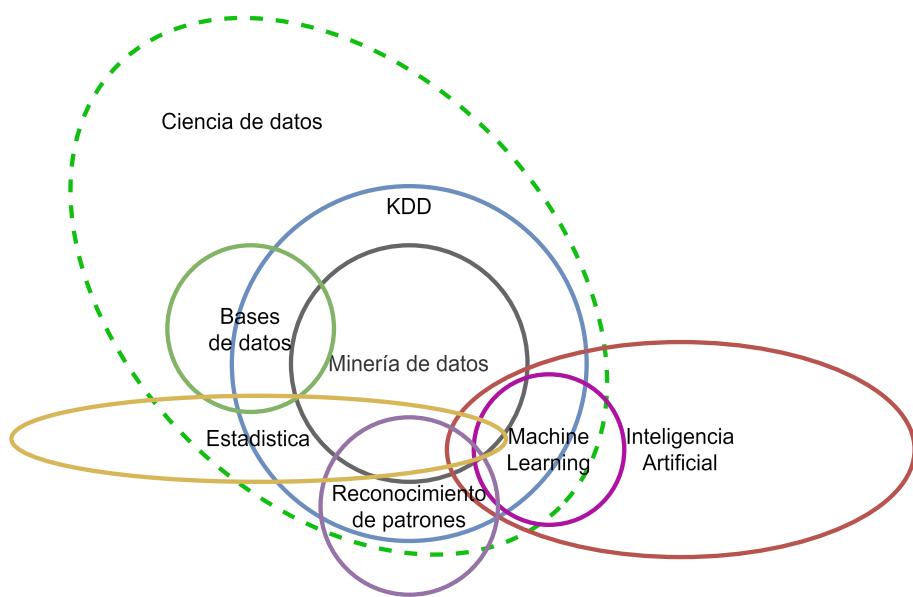


Figura 2.1: Ciencia de datos como campo interdisciplinario.
Fuente: Elaboración propia.

Podemos dividir este trabajo en dos secciones muy relacionadas, la primera la in-

teligencia artificial y de segundo punto de importancia la seguridad de la información. Desde sus inicios la inteligencia artificial, aunque con buenos resultados en muchos campos de aplicación resultaba en grandes fallos de seguridad, fiabilidad y robustez. Por cómo están entrenadas las inteligencias artificiales ([ANN](#)) tiene múltiples puntos de ataque que son susceptibles de ser atacados, los principales son los datos, las arquitecturas o los pesos. Ya que alterando cualquiera de estos componentes de forma se verá enormemente afectada el comportamiento.

2.2. Antecedentes

2.2.1. Historia, línea temporal

A continuación, se describe una línea de temporal con los hitos más relevantes en el desarrollo de las redes neuronales adjuntando con referencias las investigaciones, artículos y publicaciones realizadas. En este caso se trata de referencias en hitos de logros a nivel teórico como práctico.

TIMELINE 1: *Hitos de las redes neuronales artificiales*

- 1676 • The Chain Rule [3]
 - 1847 • Augustin-Louis Cauchy [4]
 - 1943 • Threshold Logic Unit (TLU) [5]
 - 1949 • Teoría Hebbiana
 - 1958 • Perceptron [6]
 - 1959-1960 • Adaline y Madaline [6]
 - 1965 • Multilayer Perceptron (MLP) [7]
 - 1967-1968 • Deep Learning by Stochastic Gradient Descent [8]
 - 1980's • Neuronas Sigmoidales
 - Feedforward neural network (FNN) [9]
 - Backpropagation (BP) [10, 11, 12]
 - 1985 • Boltzmann Machine [13]
 - 1987 • Adaptive resonance theory (ART) [14]
 - 1989 • Convolutional neural networks (CNN) [15]
 - Recurrent neural networks (RNN) [16]
 - 1990 • Generative Adversarial Networks (GAN) as Game [17]
 - 1997 • Long short term memory (LSTM) [18, 19]
 - 2006 • Deep Belief Networks (DBN) [20]
 - Restricted Boltzmann Machine [21]
 - Encoder / Decoder (Auto-encoder) [21]
 - 2014 • Generative Adversarial Networks (GAN) Moderns [22, 23]
 - 2018 • Style Generative Adversarial Networks (Style-GAN) [24]
-

2.2.2. Historia de la Inteligencia Artificial

Todo surge en 1676 por Gottfried Wilhelm Leibniz [2.2](#) publicó la regla de la cadena del cálculo diferencial, esencial para el análisis matemático, es la esencial para calcular como cambiará la función final si se cambian los pesos de funciones anteriores.

La regla de la cadena es fundamental para técnicas como el descenso de gradiente, propuesto por Augustin-Louis Cauchy en 1847 y utilizado para ajustar iterativamente los pesos de una NN durante el entrenamiento. Posteriormente en 1805 Adrien-Marie Legendre y Johann Carl Friedrich Gauss desarrollaron [NN](#), matemáticamente eran regresiones lineales muy simples, similares a las redes neuronales lineales simples. Esto lo uso Gauss para redescubrir el planeta enano Ceres.



Figura 2.2: Retrato de Gottfried Leibniz.
Fuente: [Wikipedia](#)

Aunque realmente la historia comienza en 1943 con la investigación de Warren McCulloch y Walter Pitts, publicaron el artículo *A logical calculus of the ideas imminent in nervous activity* [\[5\]](#). Dicho artículo creó distintas ramas de investigación (ordenadores digitales, inteligencia artificial, funcionamiento del perceptrón).



Figura 2.3: Warren Sturgis McCulloch Interview.

Fuente: [Entrevista en 1969](#)

En 1956 en la primera conferencia de inteligencia artificial organizada por la fundación Rochester, se reúnen los investigadores fundadores de los conceptos actuales de la IA (Minsky, McCarthy, Rochester, Shannon), gran parte de la bibliografía se refiere a este punto como el origen y contacto de las redes neuronales artificiales. En dicha conferencia (*Nathaural Rochester*) presento el modelo de una red neuronal que fue el resultado de la investigación desarrollada por el equipo de investigación de IBM.

1956 Dartmouth Conference: The Founding Fathers of AI

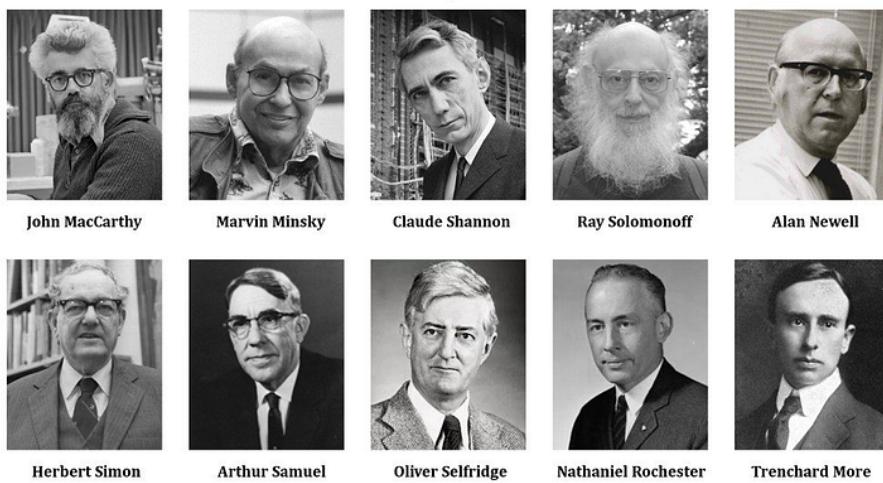


Figura 2.4: Los padres de la inteligencia artificial.

Fuente: [Linkedin](#)

En 1957 se presenta el “*Perceptron*” por Frank Rosenblatt, dicho elemento es un sistema clasificador de patrones, además contaba con la capacidad de aprender, de

ser robusto matemáticamente y poder adaptarse si algún componente se dañaba.

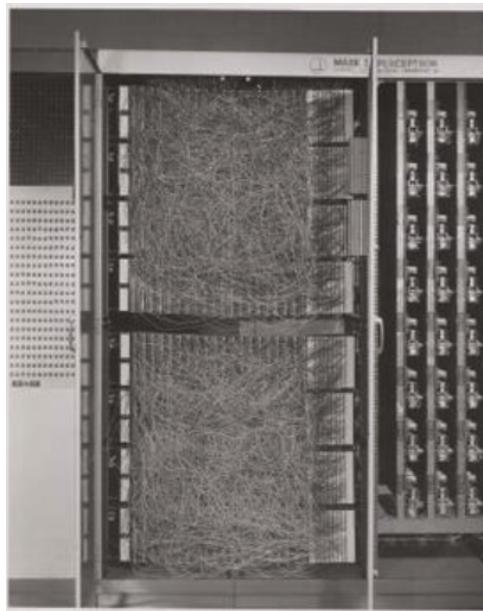


Figura 2.5: Mark I Perceptron.
Fuente: [Wikipedia](#)

El *Perceptron* fue diseñado originalmente para el reconocimiento óptico usando un sistema de 400 fotocélulas en rejilla. Posteriormente se describió el problema de no-linealidad que presentaban los perceptrones (Problema XOR) [25].

De 1959 a 1960 Bernard Widrow y Ted Hoff desarrollaron “Adaline” y “Madaline” [26] que resolvía el problema de la no-linealidad y que tenía aplicación en el reconocimiento de voz, series temporales, caracteres, etc.

Posteriormente el *MIT* realizó una investigación matemática muy crítica de todos los problemas que presentaba el *Perceptron* llegando a la conclusión que tenían grandes problemas que no podrían ser resueltos, por lo que en la próxima década (años 60) se redujo drásticamente las investigaciones sobre el campo de las redes neuronales. Esto llevó a uno de los famosos inviernos de la inteligencia artificial (1974 - 1980).



Figura 2.6: Kunihiko Fukushima.
Fuente: [IEICE](#)

Durante la década de los 70 se hacen aportes a la teoría *Hebbiana*, se aportan logros en al análisis y descripción de reglas adaptativas, además de otros aportes al principio de aprendizaje competitivo. En 1979 presentó Kunihiko Fukushima la primera red neuronal [Convolutional Neural Network \(CNN\)](#), la llamó Neocognitron [27], dicho trabajo en un futuro se mejoraría con técnicas de [Backpropagation \(BP\)](#).

En la década de los 80 se realizaron aportes como el algoritmo de [BP](#) que surgió del artículo de Hopfield [28], esto despertó la curiosidad de muchos investigadores a volver al campo de las redes neuronales. Se realizaron aportes como las redes [Graph Neural Network \(GNN\)](#). La investigación continuó con Stephen Grossberg que realizó aportes derivados de estudios fisiológicos de cómo funcionaban las neuronas y la plasticidad, lo que permitió la creación de reglas y postulados, esto se ve en los trabajos de las redes [ART](#) [14]. La investigación de Hopfield basada en el trabajo de Stephen Grossberg creó un sistema computacional neuronal interconectado que tiende a un mínimo de energía. En 1985 David E. Rumelhart basándose en la investigación realizada por Paul Werbos [11] realizó un análisis experimental del algoritmo [BP](#) y su aplicación en redes [FNN](#) [9].

Yann LeCun junto a su equipo, en 1989 crearon la primera aplicación [CNN](#) con técnicas [Backdoor Pattern \(BP\)](#) dicha aplicación podía reconocer números a partir de imágenes.



Figura 2.7: Adaptive Systems Research Department at Bell Labs 1989.
Fuente: [Twitter Yann Lecun](#)

En la década de los 90 se presentaron múltiples investigaciones y muchos avances en el campo, uno de los más importantes fue la presentación de la primera [GAN](#) como una curiosidad, ya que se presentó como un duelo entre dos redes neuronales, en un

principio fue un generador probabilístico y un predictor con el objetivo de maximizar la pérdida de cada uno en un juego *minimax*.

En 1991 se presentó el trabajo *Predictability Minimization* [29] dichas técnicas sirvieron de inspiración para el aprendizaje por refuerzo, En marzo de 1991 se hizo una aproximación a los *transformers* con auto atención, lograron separar el conocimiento del control como una máquina clásica, pero de una forma completamente neuronal, además de gestionar actualizaciones de los pesos de forma muy rápida y eficiente.

Durante la década de 1990 las redes neuronales tendían a ser muy sencillas, con pocas capas y no muy complejas por las limitaciones técnicas de la época. Por lo que muchos investigadores propusieron soluciones similares a las redes [RNN](#) que permitían una retroalimentación, además de aceptar secuencias de información arbitraria. Otros propusieron soluciones como la jerarquía de [RNN](#) autosupervisada que aprende representaciones en distintos niveles de abstracción. Comienzan a proponerse redes similares a las que en un futuro se llamarían [DBN](#) como un método no supervisado para [FNN](#).

En junio de 1991 Sepp Hochreiter Figura 2.8 implementó el primer compresor de redes neuronales, además demostró uno de los principales problemas de las [NN](#) el llamado problema del desvanecimiento o explosión del gradiente 2.3.5.3 que hacía que el aprendizaje fallará. Un análisis posterior condujo a los investigadores a una primera aproximación [LSTM](#), aunque no sería hasta 1997 con la revisión por pares y publicación del artículo *Long short-term memory* [19] que se solucionaría parcialmente el problema.

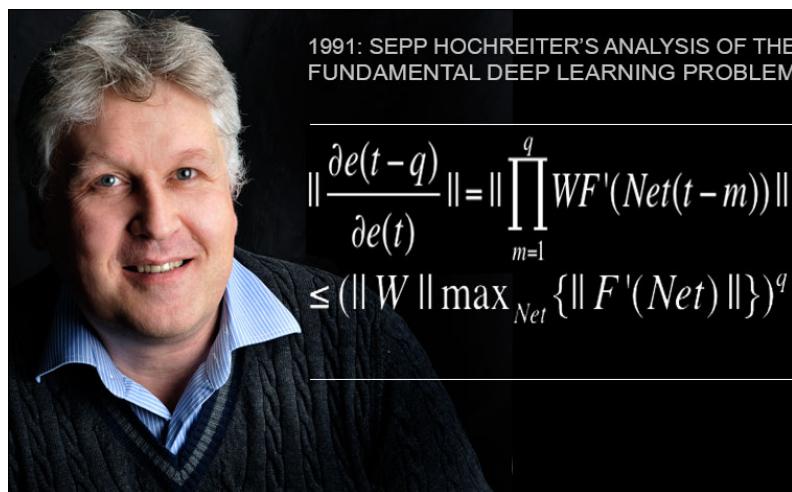


Figura 2.8: Sepp Hochreiter.
Fuente: [IDSIA](#)

Más adelante en el 2014 Goodfellow Figura 2.9 presentó la primera red neuronal

GAN pura para la generación de imágenes mediante el enfrentamiento de una red neuronal generativa contra una red neuronal discriminante entrenadas con el mismo conjunto de datos [23]. Durante los próximos años se realizaron muchos aportes a las redes neuronales generativas, principalmente de paralelización de los cálculos, técnicas de estabilización, generación condiciona, arquitecturas más eficientes, funciones de pérdidas más adecuadas, aplicaciones específicas (cambiar el estilo de pintura), redes apiladas, etc. Fruto de todo ello NVIDIA en 2018 presento [Style Generative Adversarial Networks \(StyleGAN\)](#) [24] aunque publicaron el código en 2019 con fuertes mejoras.



Figura 2.9: Ian Goodfellow.
Fuente: [MIT Technology Review](#)

2.3. Estado del arte

2.3.1. La ciencia de datos

La ciencia de datos (*Data Science*) es el estudio de los datos con el objetivo de extraer información útil, se usa principalmente para dar información útil a empresas. Es un campo multidisciplinar, ya que combina campos de las matemáticas, estadística e inteligencia artificial para analizar grandes cantidades de datos. Esto tiene como objetivo responder a las siguientes cuestiones: *¿Qué paso?, ¿Por qué pasó?, ¿Qué pasará? O ¿Qué se puede hacer con los resultados?* [30]

La ciencia de datos analiza los datos de distintas formas.

1. **Análisis descriptivo:** examina datos con visualizaciones (gráficos, tablas) para entender eventos pasados o actuales.
2. **Análisis de diagnóstico:** profundiza en los datos para entender las razones detrás del evento. Emplea técnicas como descubrimiento de datos o correlaciones.

-
- 3. **Análisis predictivo:** utiliza datos históricos y técnicas como machine learning para hacer predicciones precisas sobre patrones futuros.
 - 4. **Análisis prescriptivo:** busca la mejor respuesta para un resultado esperado. Utiliza técnicas como simulación y redes neuronales para recomendar el mejor curso de acción entre varias alternativas.

2.3.2. La minería de datos

La minería de datos, es una técnica asistida por computadora, procesa grandes conjuntos de datos para descubrir patrones y relaciones ocultas. Este conocimiento resultante se aplica en la resolución de problemas, análisis de decisiones empresariales, etc. Esta técnica tiene distintas fases para procesar y extraer información útil.

- 1. Comprender, identificar y definir el alcance del proyecto.
- 2. Comprender los datos.
- 3. Depurar datos (limpiar, integrar y dar formato).
- 4. Modelar datos.
- 5. Evaluar los resultados.
- 6. Implementar resultados.

La minería de datos es distinta en función de los datos y el objetivo, la mayoría del estado del arte segmenta la minería en tres tipos, minería de procesos, minería de textos y minería predictiva.

El Descubrimiento de Conocimientos en Bases de Datos **KDD** es un proceso que utiliza algoritmos de minería de datos para explorar y extraer conocimientos útiles de grandes bases de datos. Con el avance tecnológico, se emplean técnicas de inteligencia artificial para este propósito, con el objetivo final de obtener conocimiento de alto nivel a partir de datos de bajo nivel. La Figura 2.10 esquematiza el proceso general del **KDD**.

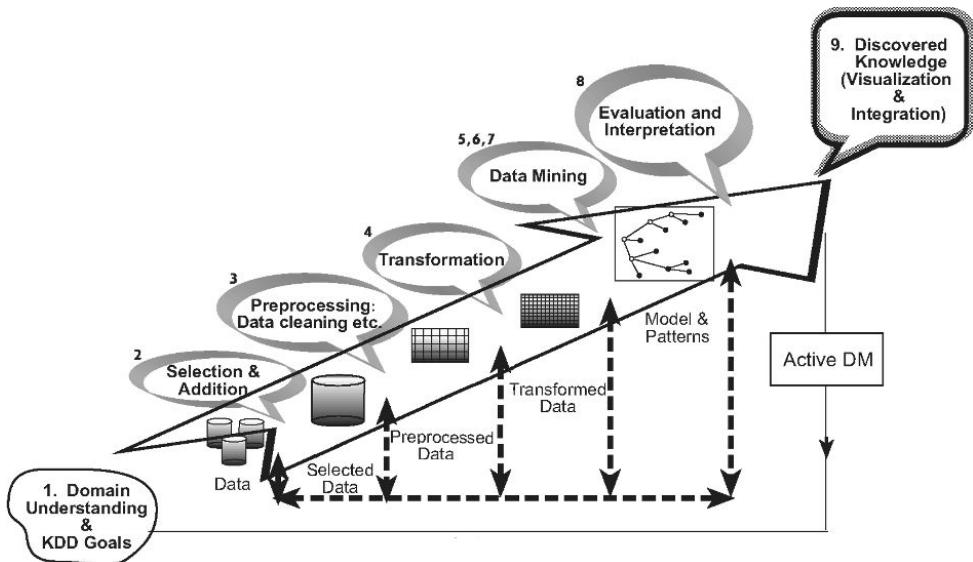


Figura 2.10: El proceso de descubrimiento de conocimiento en bases de datos.
Fuente: Sección *Introduction to Knowledge Discovery and Data Mining* [31]

1. **Comprensión del Dominio de Aplicación:** se debe comprender los datos que se tratan y el campo de obtención con el objetivo de preparar los datos.
2. **Selección y Creación de Conjunto de Datos:** se han de seleccionar los datos relevantes (Selección de características).
3. **Preprocesamiento y Limpieza de Datos:** se han de eliminar las características no relevantes, datos perturbados, entradas incoherentes o tratar datos faltantes, etc. con el objetivo de obtener datos más limpios.
4. **Transformación de Datos:** se deben tratar los datos, reduciendo la dimensionalidad, filtración, descomponiendo, etc. para que sea más sencillo el consumo de estos datos por una aplicación.
5. **Elección de Tarea de Minería de Datos:** en función de los datos y del objetivo deberemos realizar una tarea (clasificación, regresión o agrupación), ya que en la minería de datos hay dos objetivos principales, **predicción** y **descripción**.
6. **Elección del Algoritmo de Minería de Datos:** deberemos elegir entre los distintos algoritmos que se usan en la minería de datos, si buscamos precisión podemos usar redes neuronales, si buscamos explicabilidad podemos usar árboles de decisión.
7. **Implementación del Algoritmo de Minería de Datos:** emplearemos el algoritmo seleccionado ajustando los parámetros para que nos del mejor resultado.
8. **Evaluación de Patrones Minados:** debemos interpretar los resultados (reglas, confiabilidad, etc.) si no cumplen los objetivos que se buscaban en el primer paso deberemos reajustar toda la metodología, desde ajustar la selección de características a la interpretación o compresibilidad del modelo.
9. **Utilización del Conocimiento Descubierto:** por último, debemos incorporar el conocimiento a los sistemas de toma de decisiones, este es el paso más importante.

tante, ya que con él podemos medir los efectos del conocimiento obtenido. Puede suceder que una vez implementado el modelo en sistema de producción pierda eficiencia si las condiciones reales son distintas a las de la creación del modelo.

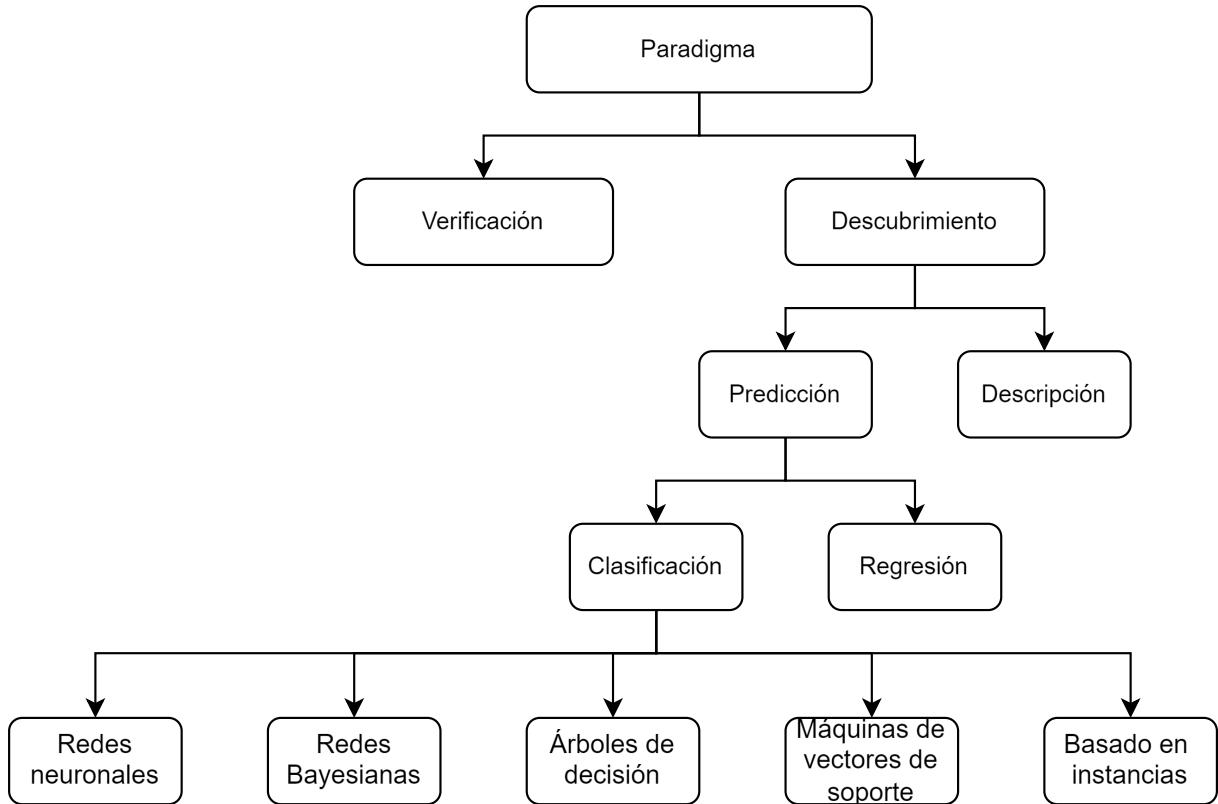


Figura 2.11: Taxonomía de minería de datos.

Fuente: Elaboración propia, inspirado en la sección *Introduction to Knowledge Discovery and Data Mining* [32]

La minería de datos puede estar orientada a la verificación o al descubrimiento de patrones, se debe automatizar su identificación, pudiendo usar un enfoque predictivo o descriptivo. Para el descubrimiento se usa el aprendizaje inductivo, mientras que en la verificación se ha de evaluar las hipótesis externas usando métodos estadísticos clásicos. La terminología clásica del aprendizaje automático está clasificado en supervisado (clasificación y regresión) y no supervisado (agrupamiento), aunque existen muchos más términos en la terminología moderna esto podemos analizarlo más en profundidad en el libro *Data Mining and Knowledge Discovery Handbook* [32].

2.3.3. Aprendizaje automático - *Machine Learning*

El *machine learning* es la ciencia o rama de la inteligencia artificial que desarrolla, modelos estadísticos, desarrolla algoritmos que generalizan comportamientos y

reconocen patrones. Es decir, hace posible el aprendizaje autónomo de las máquinas para realizar tareas sin la necesidad programar las instrucciones explícitamente. El *machine learning* busca procesar grandes cantidades de datos e identificar patrones de datos de forma automática.

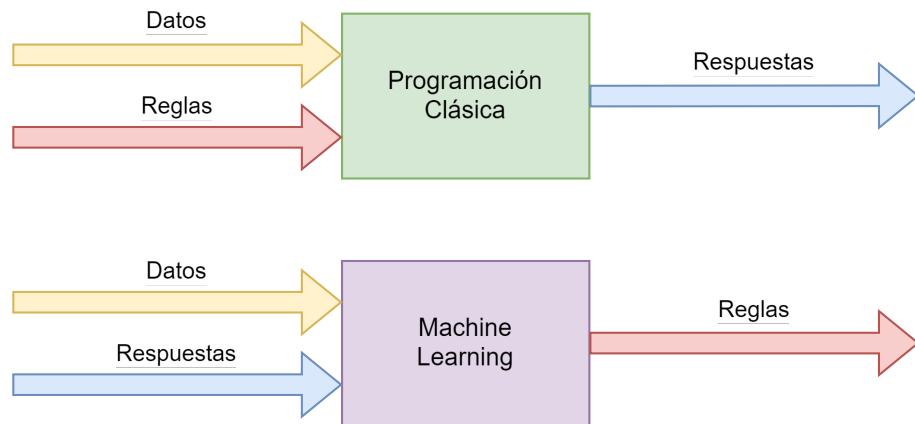


Figura 2.12: El *Machine Learning* como paradigma de programación.
Fuente: Elaboración propia.

Como podemos ver en la Figura 2.12 en el paradigma clásico se necesitaba conocer el dominio de los datos y las reglas en profundidad y programar los descriptores de forma manual para procesar los datos. En el paradigma del *machine learning* permite extraer esas reglas y patrones para procesar nuevos datos a partir de respuestas que conocíamos previamente, estas respuestas previas deben ser extraídas o validadas por expertos.

Una clasificación de las tareas del *machine learning* lo podemos ver en la Figura 2.13, esta clasificación está dividida en función de cómo se realiza el aprendizaje del modelo.

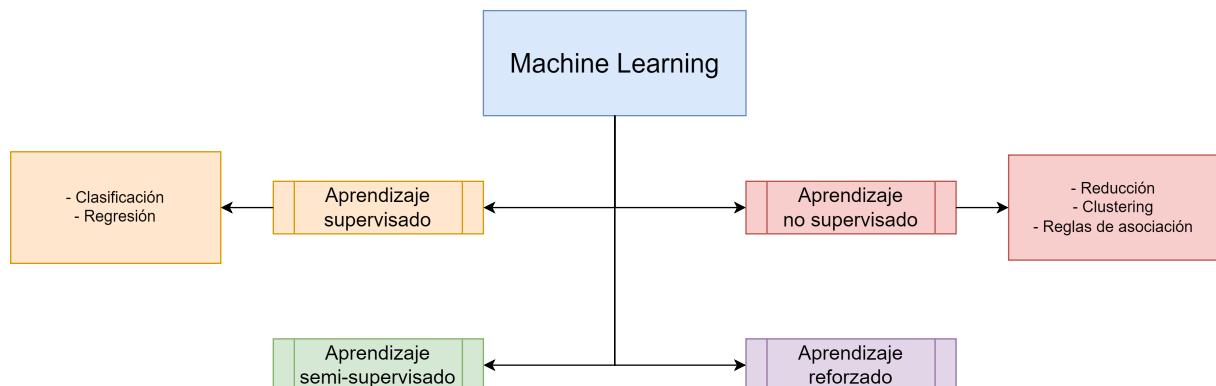


Figura 2.13: Clasificación de los algoritmos de aprendizaje automático.
Fuente: Elaboración propia.

El *machine learning* puede trabajar con datos estructurados y no estructurados,

aunque estos últimos requieren un procesamiento previo para adaptarlos en un formato estructurado.

2.3.4. Aprendizaje profundo - *Deep Learning*

El *deep learning* es una subrama del *machine learning* que usa redes neuronales artificiales para analizar datos no lineales, estas redes imitan el comportamiento del cerebro humano, para esto suelen contar con arquitecturas complejas. Se distingue del *machine learning* por los tipos de datos con los que puede trabajar y por los métodos con los que hace que los modelos aprendan.

Otra de las características que diferencia al *deep learning* del *machine learning* es que elimina parte del procesamiento previo de datos, ya que sus algoritmos pueden ingerir y procesar datos no estructurados.

2.3.5. Redes neuronales artificiales - *Artificial Neuronal Networks*

El *deep learning* emplea principalmente redes neuronales para reconocer, clasificar y describir con precisión patrones de datos. Estas redes están inspiradas en el funcionamiento del cerebro humano. [33]

Las redes neuronales es un conjunto de neuronas artificiales que están organizadas generalmente por capas. Estas redes tienen una capa de entrada, una capa de salida y múltiples capas ocultas con distintas funciones de activación que ponderan los pesos de cada neurona.

Existen formas muy variadas de componer las capas, a esto se le denomina **topología de red neuronal**, en función del objetivo que se busque, la topología será muy distinta.

2.3.5.1. Elementos de una neurona artificial

Como se ha comentado previamente las neuronas artificiales están inspiradas en las neuronas biológicas del cerebro humano, fueron los investigadores McCulloch y Pitts [5] los que sentaron las bases de lo que es una neurona artificial, demostrando que su modelo de red neuronal podía realizar cálculos que se correspondían con la lógica proposicional.

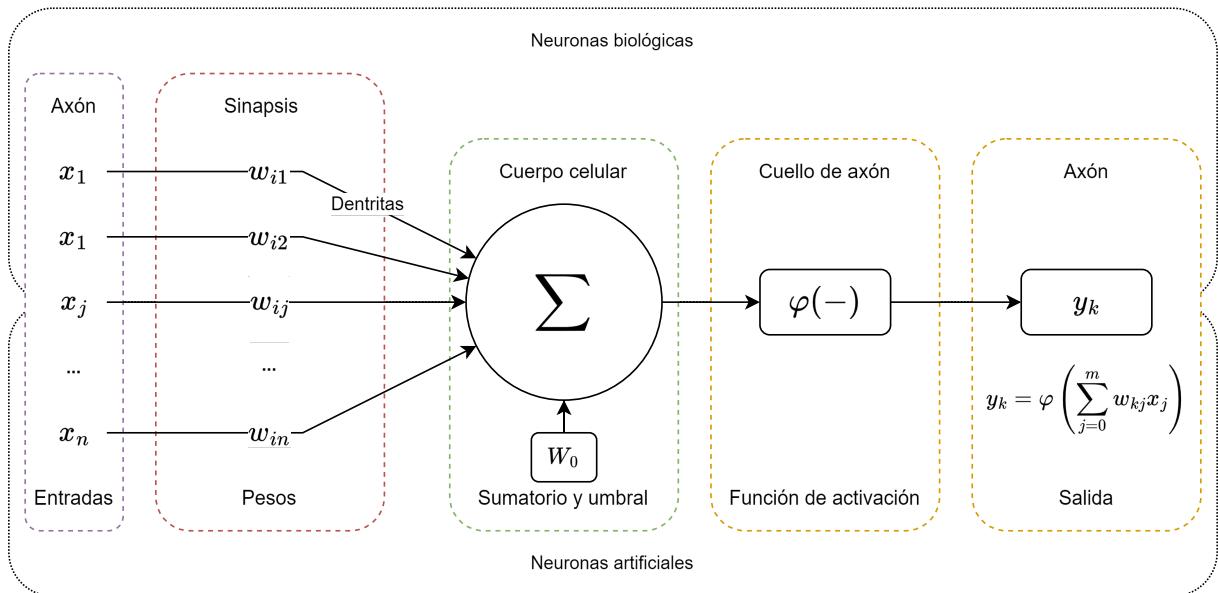


Figura 2.14: Partes de una neurona artificial bioinspirada en una neurona biológica.
Fuente: Elaboración propia

En la Figura 2.14 podemos ver las distintas partes de una única neurona artificial y sus partes equivalentes con una neurona biológica. El axón opera como la entrada de información a la neurona, la sinapsis como las conexiones con la neurona que son los pesos, el cuerpo celular como la neurona que representa la unidad de procesamiento central donde se pondera la suma de las señales de entrada, el cuello del axón sería nuestra función de activación que se activa cuando supera un umbral y el axón sería nuestra señal de salida.

2.3.5.2. Funcionamiento de una red neuronal

Las redes neuronales profundas tienen multitud de capas con nodos interconectados, cada capa sobre la capa anterior con el objetivo de optimizar la precisión de una predicción o clasificación. A esta progresión de cálculos se le denomina “propagación hacia delante”.

El proceso de “propagación inversa” es el encargado de calcular errores de precisión, ajustar sesgos y ponderaciones usando algoritmos como [El descenso de gradiente](#).

La capa de entrada es por donde el modelo ingiere los datos, y la capa de salida es donde el modelo responderá con la predicción o clasificación una vez se haya completado la fase de aprendizaje.

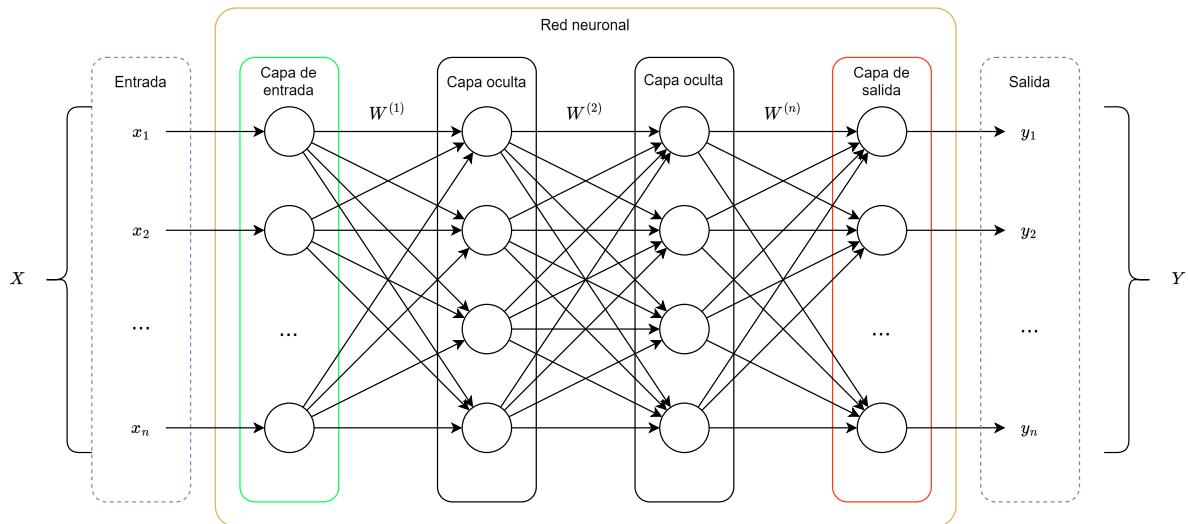


Figura 2.15: Red neuronal artificial.
Fuente: Elaboración propia

En la Figura 2.15 podemos ver una arquitectura simple de una red neuronal que cuenta con la capa de entrada, la capa de salida y dos capas ocultas.

El vector de entrada $X = (x_1, x_2, \dots, x_n)$ será la información suministrada a nuestra red a través de la capa de entrada, este vector será de tamaño n .

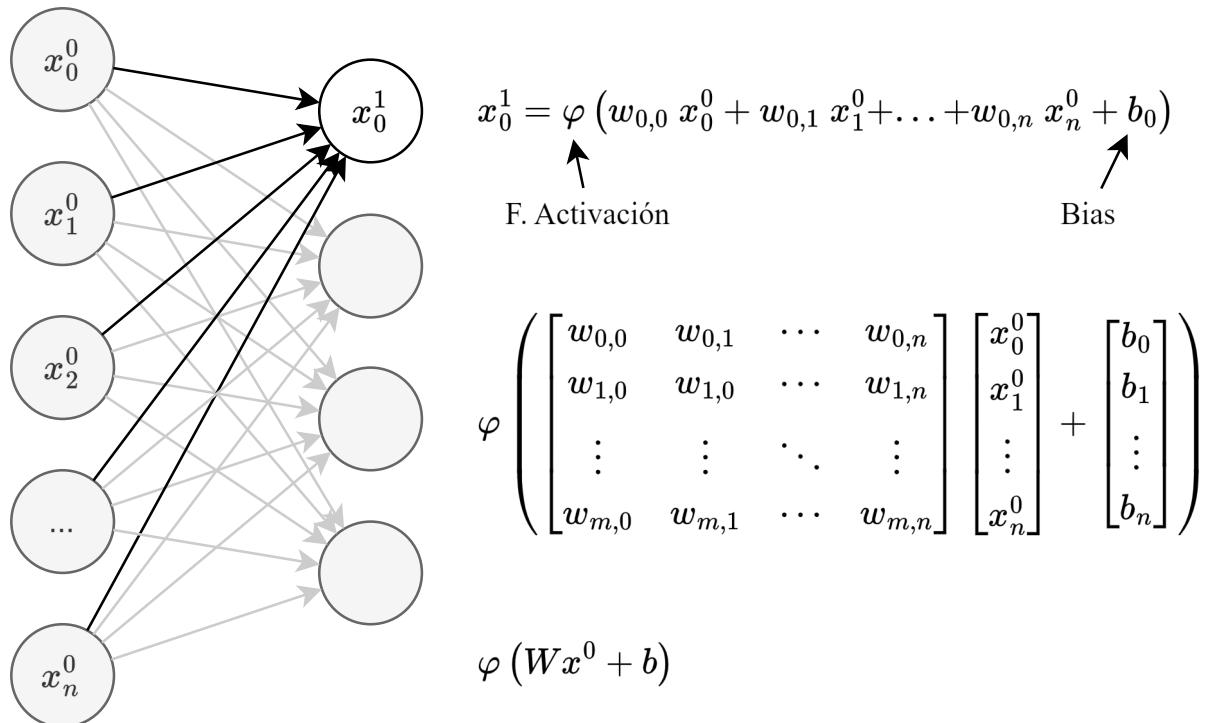


Figura 2.16: Cálculo de los pesos de la red neuronal artificial.
Fuente: Elaboración propia

1. Función de la red neuronal: $f(x) = Wx + b$
2. Matriz de pesos: $W = [w_{0,n}, w_{1,n}, \dots, w_{m,n}]$

-
3. Vector de pesos: $w_n \in \mathbb{R}^D$
 4. Vector de entrada: $x \in \mathbb{R}^D$
 5. Vector del bias: $b \in \mathbb{R}^C$

El perceptrón simple

El Perceptrón es una neurona artificial simple llamada también como [Linear Threshold Unit \(LTU\)](#). La [LTU](#) calcula una suma ponderada de los valores de las entradas $Z = w_1x_1 + w_2x_2 + \dots + w_nx_n$ se aplica el umbral para generar el resultado.

La función umbral común en [LTU](#) es la función heaviside [2.1](#).

$$heaviside(z) = \begin{cases} 0 & \text{si } z < 0 \\ 1 & \text{si } z \geq 0 \end{cases} \quad sgn(z) = \begin{cases} -1 & \text{si } z < 0 \\ 0 & \text{si } z = 0 \\ +1 & \text{si } z > 0 \end{cases} \quad (2.1)$$

Se puede emplear para una clasificación binaria lineal simple, mediante una combinación lineal de las entradas; si la operación supera el umbral, se obtendrá un resultado binario en un vector.

El algoritmo de entrenamiento del perceptrón se basa en ajustar los pesos de conexión entre las entradas y la neurona de salida para mejorar la precisión de las predicciones. Está inspirado en la regla de Hebb, que sugiere que las conexiones entre neuronas se fortalecen cuando una neurona activa repetidamente a otra.

Primero inicializa los pesos, predice, evalúa su error y actualización de los pesos para en la siguiente iteración minimizar su error.

Rosenblatt demostró que el algoritmo convergería hacia una solución, esto es llamado el Teorema de convergencia del perceptrón [\[34\]](#).

Marvin Minsky y Seymour encontraron una serie de debilidades que los perceptrones son incapaces de resolver, algunas de estas debilidades se pueden resolver apilando varios perceptrones, la [Artificial Neural Network \(ANN\)](#) resultante se le llama [Multilayer Perceptron \(MLP\)](#).

Uno de los problemas clásicos que tenían los perceptrones simples era el problema de clasificación XOR, esto está representado en la Figura [2.17](#). Como tal el perceptrón es un modelo que puede aprender funciones lineales, pero el problema XOR no es linealmente separable, lo que significa que no se puede trazar una línea recta para separar las dos clases (A y B) en el espacio de entrada binario de dos dimensiones.

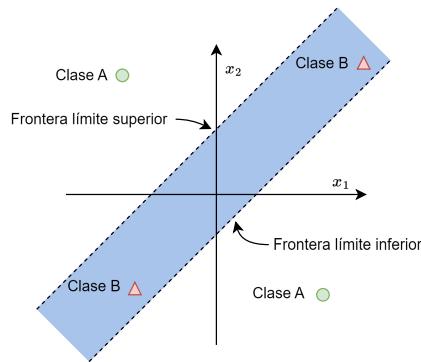


Figura 2.17: Problema de clasificación XOR.
Fuente: Elaboración propia

El perceptrón multicapa y retropropagación

Fue en 1986 cuando D. E. Rumelhart propuso la retro propagación junto a múltiples capas para resolver el problema de clasificación XOR.

El proceso es el siguiente, por cada instancia de entrenamiento se calcula la salida de cada neurona en la capa consecutiva (paso hacia adelante). A continuación se mide el error de salida de la red (la diferencia entre la salida deseada y la de la red) calculará cuánto contribuyó cada neurona en la última capa oculta al error de cada neurona de salida. Se mide cuantas contribuciones de error provienen de cada neurona en la capa oculta anterior, así sucesivamente hasta que se llega a la capa de entrada (paso hacia atrás). Se mide eficientemente el gradiente de error en todos los pesos de la conexión en la red propagando el gradiente de error hacia atrás. [34]

Este proceso funciona, ya que se cambió la función escalonada de las LTU a una función logística Sigmoid, esto se requería, ya que la función de paso contiene solo segmentos planos y por tanto no había gradientes. Esto permitió que el descenso de gradiente pudiera ir progresando lentamente hasta converger. El algoritmo de retroalimentación se puede usar con otras funciones de activación que veremos más adelante.

2.3.5.3. El descenso de gradiente

El método de descenso de gradiente trata de encontrar un mínimo en una función dada, ya sea local o global. El método usa el gradiente negativo $-\nabla f$, con este método obtenemos la dirección con el descenso máximo en los valores de la función, con esto buscamos encontrar la posición mínima.

En términos simples, el descenso de gradiente va ajustando iterativamente los valores de los parámetros en la dirección opuesta al gradiente de la función de costo. Al moverse en la dirección opuesta al gradiente, el algoritmo busca alcanzar el mínimo local o global de la función de costo.

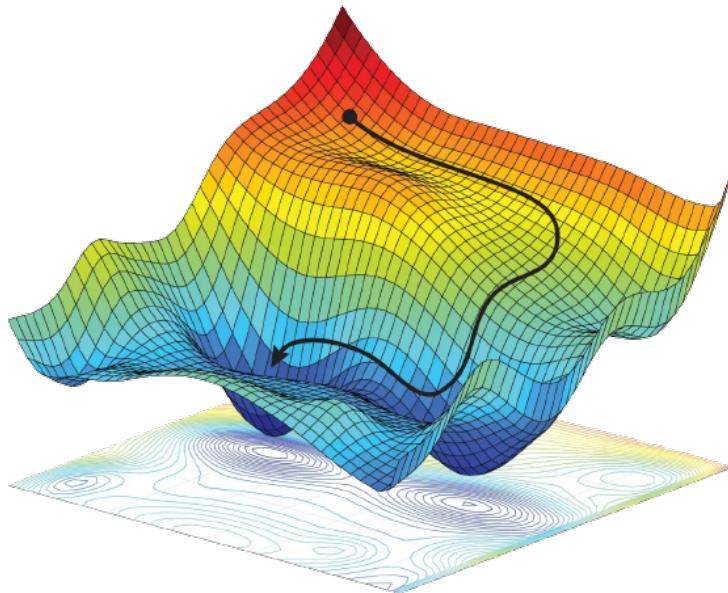


Figura 2.18: Descenso de gradiente
Fuente: [F\(x\) Data Labs Pvt. Ltd.](#)

Algoritmo 1: Representación del descenso de gradiente

```
1 Seleccionar  $x^{(0)}$ 
2 Establecer  $k \leftarrow 0$ 
3 while  $\|\nabla f(x^{(k)})\| \geq \epsilon$  do
4    $x^{(k+1)} = x^k - t_k - t_k \nabla f(x^{(k)})$ 
5    $k \leftarrow k + 1$ 
6 end
7 return  $x^{(k)}$ 
```

Hay tres variantes del algoritmo de aprendizaje de descenso de gradiente [35].

- **Descenso de gradiente por lotes:** suma el error para cada punto en un conjunto de entrenamiento, actualiza el modelo después de que todos los ejemplos de entrenamiento han sido evaluados, esto es lo que se le conoce como épocas. Es un método eficiente computacionalmente, pero costoso para grandes conjuntos de datos, ya que requiere almacenar en memoria todos los datos, suele producir un gradiente de error estable y una convergencia, pero a veces ese punto de convergencia no es el ideal y encuentra el mínimo local frente al global.
- **Descenso de gradiente estocástico:** en cada época actualiza los parámetros del

ejemplo de entrenamiento, esto reduce la necesidad de memoria, la actualización proporciona velocidad y detalles, aunque requieren un coste computacional mayor. Los gradientes son más ruidosos por sus actualizaciones frecuentes, esto le permite salir de mínimos locales.

- **Descenso de gradiente por mini lotes:** combina aspectos de del descenso de gradiente por lotes como el estocástico. Divide el conjunto de entrenamiento en pequeños lotes y realiza actualizaciones en cada uno de estos lotes. Esto permite tener un equilibrio entre eficiencia computacional y velocidad del descenso de gradiente.

Aspecto	Descenso de Gradiente por Lotes	Descenso de Gradiente Estocástico	Descenso de Gradiente por Mini Lotes
Eficiencia computacional	Eficiente para conjuntos de datos pequeños.	Menos eficiente para grandes conjuntos de datos.	Equilibrio entre eficiencia y velocidad.
Coste computacional	Puede ser lento para grandes conjuntos de datos.	Más rápido que por lotes, pero aún puede ser lento.	Mejor tiempo de procesamiento que por lotes.
Coste en memoria	Requiere almacenar todo el conjunto en memoria.	Solo necesita almacenar un ejemplo a la vez.	Menos demandante que por lotes, pero eficiente.
Convergencia del Modelo	Propenso a converger hacia mínimos locales.	Puede saltar mínimos locales pero es ruidoso.	Convergencia más estable que SGD, menos ruido.
Sensibilidad a datos ruidosos	Menos sensible debido al promedio de errores.	Más sensible debido a actualizaciones frecuentes.	Menos sensible que SGD, gracias al promedio.
Tamaño del conjunto de datos	Funciona bien para conjuntos de datos pequeños.	Buen rendimiento para distintos tamaños.	Equilibrio, adecuado para diferentes tamaños.
Precisión del modelo	Convergencia más estable, pero a veces sub óptima.	Puede converger más rápidamente, pero menos estable.	Balance entre estabilidad y rapidez.

Tabla. 2.1: Tabla comparativa de los tipos de descenso de gradiente

Problema de desvanecimiento y explosión de gradiente

Hablaremos primero del desvanecimiento de gradiente, esto ocurre durante la retropropagación, ya que al movernos hacia atrás el gradiente sigue haciéndose más pequeño, esto hace que las capas anteriores de la red aprendan más lentamente que las posteriores. A medida que el algoritmo itera los cambios se hacen insignificantes lo que hace que el algoritmo se estanque.

El problema de la explosión de gradiente es lo contrario, el gradiente se hace demasiado grande creando un modelo inestable. Los pesos del modelo crecerán y eventualmente se representarán como NaN.

2.3.5.4. Optimizadores

Los optimizadores son los algoritmos que buscan encontrar la mejor solución para un problema esto se utilizan en conjunto con el descenso de gradiente para mejorar su eficiencia y rendimiento, generalmente minimizando o maximizando una función objetiva al ajustar sus parámetros. Existen muchos optimizadores que nos permiten explorar de forma eficiente el espacio de posibles soluciones.

Estos son algunos de los optimizadores más famosos.

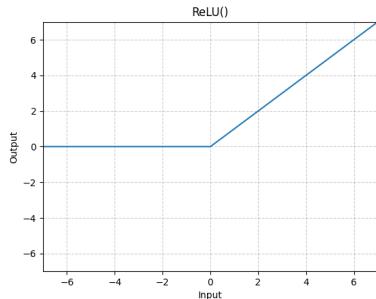
1. **SGD** (Stochastic Gradient Descent with Momentum): Descenso de gradiente estocástico con momentum.
2. **Adam** (Adaptive Moment Estimation): Descenso de gradiente con tasa de aprendizaje adaptativa.
3. **Adamax** (Adaptive Moment Estimation with Infinity Norm): variante de Adam que utiliza la norma infinita en lugar de la norma 2 para calcular y actualizar el término de escala máximo.
4. **AdaGrad** (Adaptative Gradient Algorithm): variante de *SGD* en la que se emplean distintas tasas de aprendizaje teniendo en cuenta el gradiente acumulado en cada una de las variables.
5. **RMSprop** (Root Mean Square Propagation): variante de *AdaGrad* en la que, en lugar de mantener un acumulado los gradientes, se utiliza el concepto de “ventana” para considerar los gradientes más recientes.

2.3.5.5. Funciones de activación

A la salida de una neurona artificial debe existir un filtro o umbral que modifica el resultado previo, esta función puede imponer un umbral que debe pasar para que el valor se transmita a la siguiente capa, esta función se le conoce como función de activación. Esto introduce no linealidad en el modelo, permitiendo que la red aprenda patrones complejos y mejora su capacidad de representación.

Existen muchas funciones de activación que modifican la red neuronal de distintas formas. Se suelen clasificar en dos categorías, funciones de activación de suma ponderada, no lineales u otras.

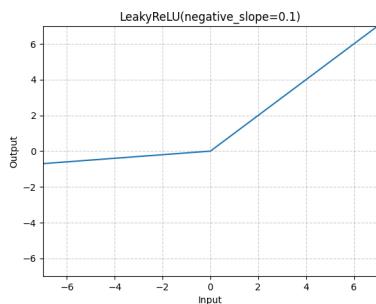
F. Activaciones no lineales - *Non-linear Activations (weighted sum, nonlinearity)* [2]



(a) Función de activación ReLU.
Fuente: [Pytorch torch.nn.ReLU](#)

$$\varphi(x) = \max(0, x)$$

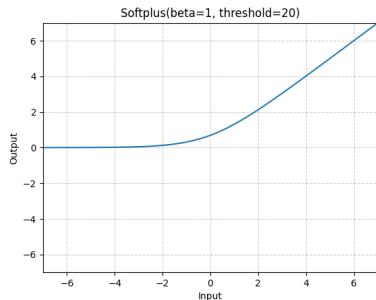
(b) Función de activación ReLU.
Fuente: [Pytorch torch.nn.ReLU](#)



(c) Función de activación LeakyReLU.
Fuente: [Pytorch torch.nn.LeakyReLU](#)

$$\varphi(x) = \max(0, x) + \text{negative_slope} * \min(0, x)$$

(d) Función de activación LeakyReLU.
Fuente: [Pytorch torch.nn.LeakyReLU](#)

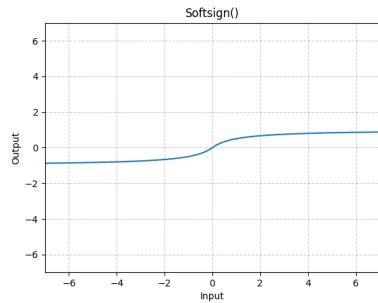


(e) Función de activación Softplus.
Fuente: [Pytorch torch.nn.Softplus](#)

$$\varphi(x) = \frac{1}{\beta} \log(1 + e^{\beta x})$$

(f) Función de activación Softplus.
Fuente: [Pytorch torch.nn.Softplus](#)

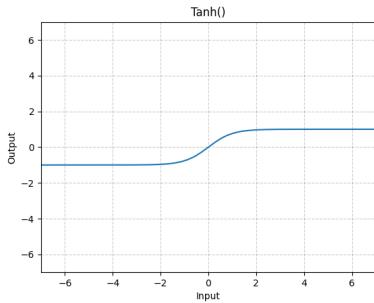
Figura 2.19: Funciones de activación no lineales



(a) Función de activación Softsign.
Fuente: [Pytorch torch.nn.Softsign](#)

$$\varphi(x) = \frac{x}{1 + |x|}$$

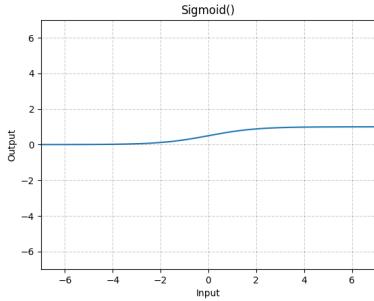
(b) Función de activación Softsign.
Fuente: [Pytorch torch.nn.Softsign](#)



(c) Función de activación Tanh.
Fuente: [Pytorch torch.nn.Tanh](#)

$$\varphi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

(d) Función de activación Tanh.
Fuente: [Pytorch torch.nn.Tanh](#)



(e) Función de activación Sigmoid.
Fuente: [Pytorch torch.nn.Sigmoid](#)

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

(f) Función de activación Sigmoid.
Fuente: [Pytorch torch.nn.Sigmoid](#)

Figura 2.20: Funciones de activación no lineales

F. Activaciones no lineales - *Non-linear Activations (other)* [2]

1. **Softmax:** Aplica la función Softmax a un Tensor de entrada n -dimensional reescalándolos para que los elementos del Tensor de salida n -dimensional se encuentren en el rango $[0,1]$. [2]

$$\varphi(x_i) = \frac{e^{x_i}}{\sum_j e_j} \quad (2.2)$$

2.3.5.6. Tipos de capas

Existen una gran variedad de capas con distintos propósitos, permiten organizar y estructurar el funcionamiento de la red durante el proceso de aprendizaje. Cada capa influye con un propósito específico, contribuyendo de manera única al funcionamiento global de la red.

Algunas de las características de las capas más famosas son las siguientes.

- **Capas densas y dispersas (Denses and Sparses Layers):**

- **Capas densas:** Capas que conectan todas las neuronas de una capa con todas las de la capa siguiente.
Son ampliamente utilizadas en [ANN](#) tradicionales para tareas de clasificación y regresión.
- **Sparses Layers:** Capas que contienen conexiones dispersas entre neuronas, lo que implica que no todas las neuronas están conectadas entre sí.
Se utilizan para reducir la complejidad computacional y el consumo de memoria en modelos grandes.

- **Capas lineales (Linear Layers):**

- Estas capas generalmente se combinan con funciones de activación no lineales para introducir no linealidad en la red neuronal.
- Tanto `nn.Linear` como `nn.Bilinear` realizan transformaciones lineales de las entradas ponderando los pesos. `nn.Identity` no realiza ninguna transformación, y `nn.LazyLinear` puede realizar transformaciones lineales perezosas.
- Proporcionan flexibilidad al permitir ajustar sus parámetros según las necesidades del modelo o la tarea.

- **Capas Convolucionales (Convolutional Layers):**

- Aplican filtros o convoluciones a la entrada para detectar patrones como bordes, texturas y formas.
- Se utilizan principalmente para procesar datos de imágenes y extraer características relevantes.
- Son la base del funcionamiento en tareas de clasificación de imágenes o de segmentación de imágenes.

- **Capas de Pooling (Pooling Layers):**

- Comprimen la dimensionalidad de las características extraídas de las capas convolucionales. Se usan entre capas convolucionales.
- Existen múltiples variantes como `nn.MaxPool1d` o `nn.AvgPool1d` que seleccionan los valores más significativos de una región dada.
- Ayudan a reducir el overfitting, simplificar la representación de características.

ticas, reducir el coste computacional, etc.

- **Capas de Padding (Paddings Layers):**

- Se usan para ajustar las dimensiones de los datos de entrada, permiten después aplicar operaciones.
- Añade ceros o constantes alrededor de los bordes de regiones para ajustar el tamaño a posteriores operaciones.
- Previenen la pérdida de información en los bordes.

- **Capas de Normalización (Normalization Layers):**

- Realizan operaciones sobre los mapas de activación, se utilizan para estabilizar y acelerar el entrenamiento de la red.
- Algunas de las capas relevantes son `nn.BatchNorm1d` y `nn.LayerNorm`, estas permiten aplicar una normalización por lotes o por capas respectivamente.
- Ayudan a mantener la distribución de activaciones más consistentes durante el entrenamiento.

- **Capas de Dropout (Dropout Layers):**

- Se desactivan aleatoriamente un porcentaje de neuronas en la capa esto evita dependencias en el entrenamiento, además de generalizar mejor.
- Es una técnica de regularización que reduce el overfitting, se usan para entrenamientos más estables eliminando dependencias.

- **Capas Recurrentes (Recurrent Layers):**

- Diseñadas para manejar datos secuenciales o de series temporales.
- Incorporan conexiones cíclicas que les permiten propagar información de un paso de tiempo a otro, las más usadas son `nn.RNN`, `nn.LSTM` y `nn.GRU`.
- Las capas recurrentes tienen memoria y contexto considerando los estados previos, son útiles para traducción de idiomas, procesamiento del lenguaje natural y reconocimiento de voz.

2.3.5.7. Topologías de redes neuronales

La topología o arquitectura de una red neuronal nos referimos a la organización y disposición de las neuronas en la red, formando capas. Estos parámetros fundamentales determinan cómo se estructura la red. Se suele considerar el número de capas, número de neuronas por capa, tipo de conexiones (unidireccionales o recurrentes).

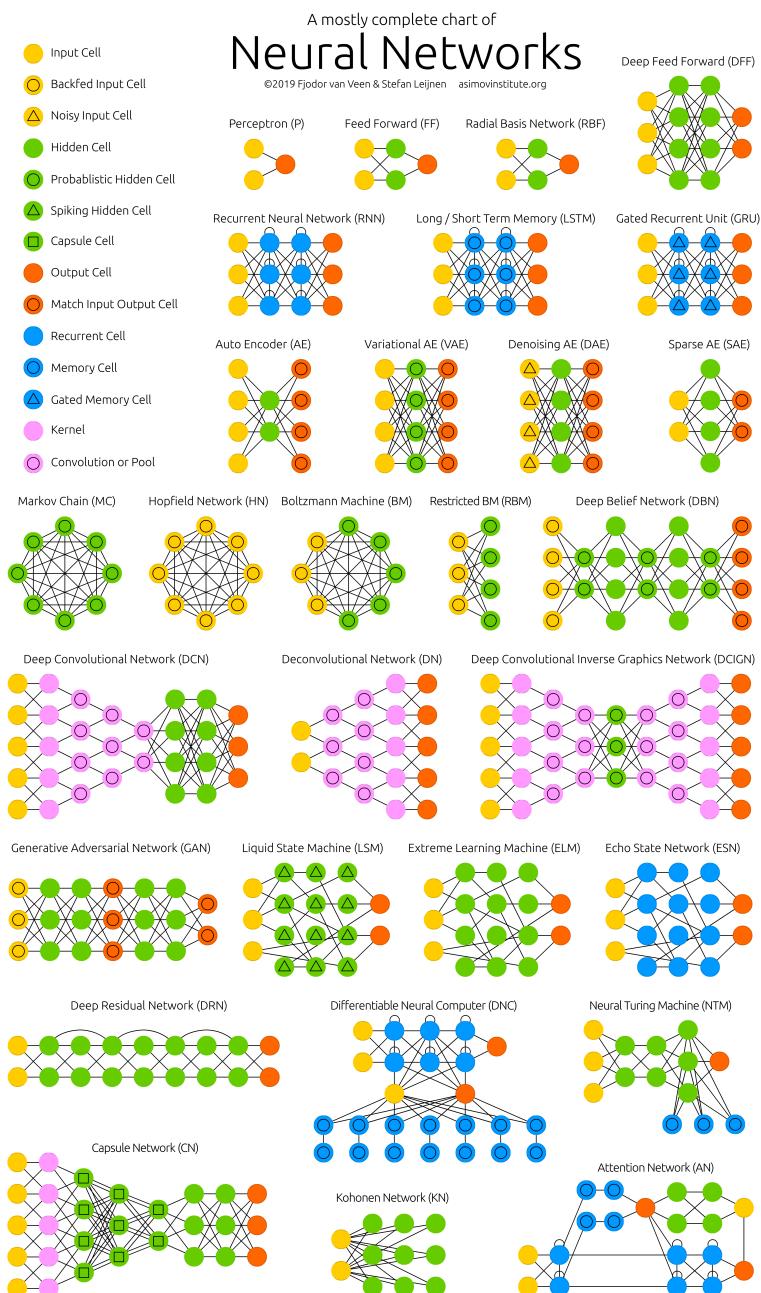


Figura 2.21: Topologías de redes neuronales.
Fuente: [The neural network zoo](http://neuralfactory.com/the-neural-network-zoo)

2.3.6. Redes generativas adversariales - *Generative Adversarial Networks (GAN)*

La arquitectura **Generative Adversarial Network (GAN)** está formada por dos redes neuronales artificiales, una “generadora (G)” y otra “discriminadora (D)”. La red G se encarga de generar instancias del mismo dominio que el conjunto de datos, mientras que la red D es la encargada de aceptar o rechazar si los datos generados por la red G son reales o falsos. Ambas redes se entrenan conjuntamente de manera que G minimiza las detecciones de D , y a su vez D detecta las instancias generadas por G [36].

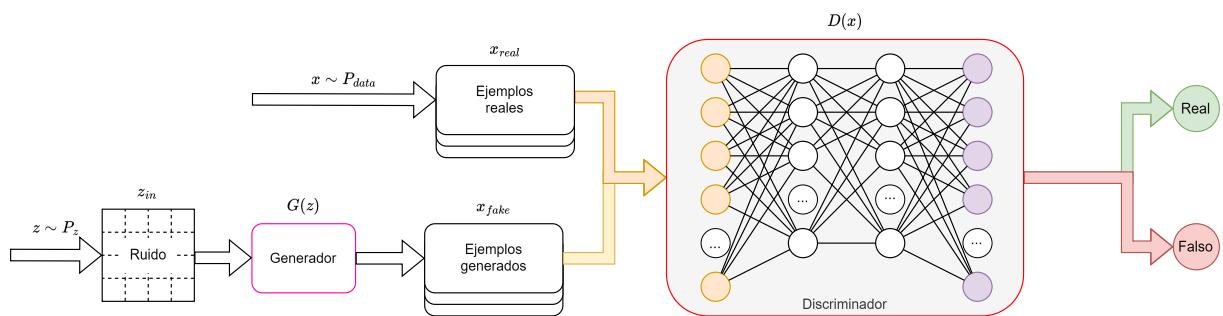


Figura 2.22: Arquitectura de las redes neuronales generativas adversariales.
Fuente: Elaboración propia.

Esto está representado en la Figura 2.22, donde podemos observar las distintas partes de una red **GAN**. Además podemos observar la distribución de probabilidad aleatoria representada como p_z . El objetivo de la red es la optimización de la distribución de probabilidad de la red generativa P_{data} sea similar a $G(z)$, es decir $G(z) \sim p_g$.

$$\min_G \max_D \mathbb{E}_{x \sim p_r} \log [D(x)] + \mathbb{E}_{z \sim p_z} \log [1 - D(G(z))] \quad (2.3)$$

La función objetiva está definida por la ecuación 2.4, nos referimos con θ a los parámetros de la red G y con ω a los parámetros de la red D . La esperanza de $f(x)$ con respecto a la distribución de probabilidad de x según Q es lo que se denota como $\mathbb{E}_{x \sim Q}$.

$$\min_{\theta} \max_{\omega} \mathbb{E}_{x \sim Q} \log [D_{\omega}(x)] + \mathbb{E}_{z \sim p_{\theta}} \log [1 - D_{\omega}(G_{\theta}(z))] \quad (2.4)$$

Para que la red D funcione deberá recibir tanto datos originales x_{real} como datos generados por G , x_{fake} , G se optimiza a la misma vez para que D no pueda detectar los datos generados por G , este proceso está representado en las ecuaciones 2.5 y 2.6.

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^m \left[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))) \right] \quad (2.5)$$

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)}))) \quad (2.6)$$

Este proceso “mín máx” hace que la red G mejore la generación de instancias haciéndola más parecidas, aunque también puede generar instancias irreales que aunque superan la validación del discriminador no son físicamente viables en un contexto real.

Las [GAN](#) básicas utilizan una clasificación binaria típica en la red D .

Usos y aplicaciones

Las redes [GAN](#) se han usado principalmente para la generación de imágenes, aunque puede generar instancias sintéticas de muchos dominios, previamente existían los modelos generativos [Variational Autoencoder \(VAE\)](#).

- Síntesis de imágenes
- Imagen a imagen
- Re escalado de imágenes
- Detección de desenfoque
- Detección de manipulación de la cámara
- Codificación de vídeo

Ventajas e inconvenientes

Semi-supervised Generative Adversarial Network (SGAN)

En las redes [Semi supervised Generative Adversarial Networks \(SGAN\)](#) [37], el discriminador cuenta con datos semi supervisados, se realizan clasificaciones con la función `nn.Softmax` para optimizar el discriminador. El uso de estos datos semi supervisados y la mejora del clasificador o predictor en la red D mejoran los resultados de ambas redes.

Conditional Generative Adversarial Network (cGAN)

Las redes [Conditional Generative Adversarial Networks \(cGAN\)](#) [38, 39, 40]

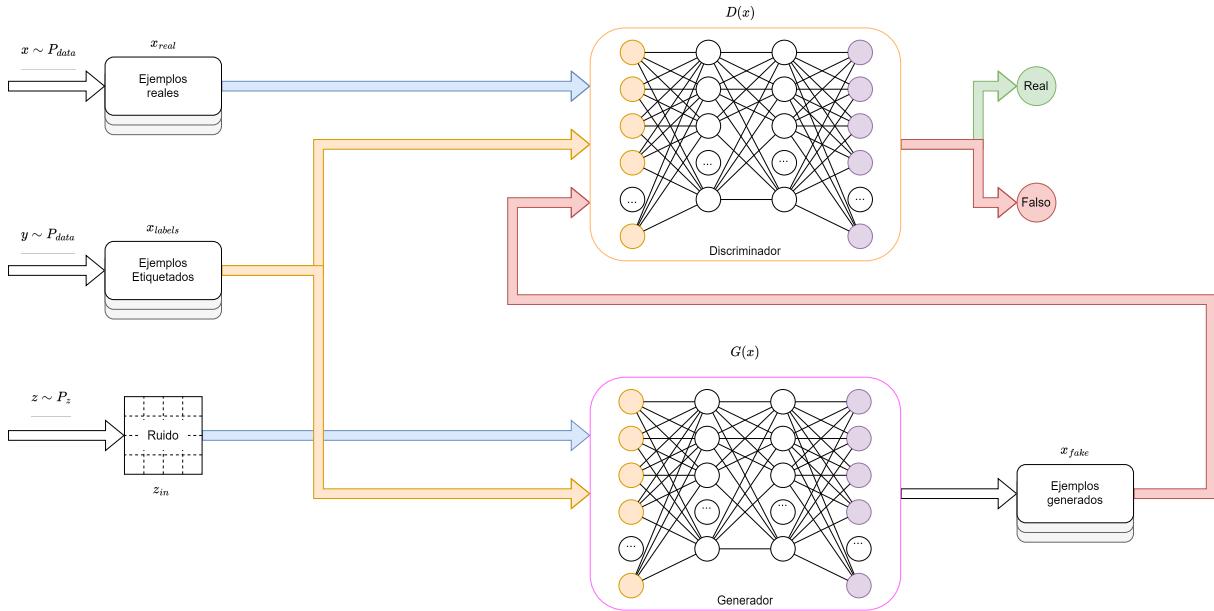


Figura 2.23: Arquitectura Conditional GAN (cGAN)
Fuente: Elaboración propia

Couple Generative Adversarial Network (CoGAN)

Cycle Generative Adversarial Network (CycleGAN)

Las redes [Cycle Generative Adversarial Networks \(CycleGAN\)](#) [41] es un método que puede extraer las características de un dominio y descubre como incorporar esas características a otro dominio con la ventajas de que no necesita de datos de entrenamiento emparejados.

[CycleGAN](#) es muy similar a [Pixel to Pixel \(Pix2Píx\)](#) [?] que a su vez deriva de las redes [cGAN](#)

Deep Convolutional Generative Adversarial Network (DCGAN)

Las redes [Deep Convolutional Generative Adversarial Networks \(DCGAN\)](#) [42, 43, 44, 45] son una de las redes generativas más famosas y usadas dentro de su ámbito

Estas redes generan detalles de alta resolución partiendo de puntos espacialmente locales de un mapa de características reducido.

El discriminador es un clasificador de imágenes basado en [CNN](#), frente al generador que a partir de una semilla (ruido) generara imágenes apilando distintas capas

Least Square Generative Adversarial Network (LSGAN)

[Least Square Generative Adversarial Networks \(LSGAN\)](#) [\[46\]](#)

Progresive Generative Adversarial Network (ProGAN)

[Progresive Generative Adversarial Networks \(ProGAN\)](#) [\[47\]](#)

Super Resolution Generative Adversarial Network (SRGAN)

Las redes [Super Resolution Generative Adversarial Networks \(SRGAN\)](#) [\[48\]](#) son una variante de las redes [DCGAN](#)

$$\min_{\theta_G} \max_{\omega_D} \mathbb{E}_{I^{HR} \sim P_{train}(I^{HR})} \left[\log D_{\omega_D}(I^{HR}) \right] + \mathbb{E}_{I^{LR} \sim P_G(I^{LR})} \left[\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))) \right] \quad (2.7)$$

Style Generative Adversarial Network (StyleGAN)

[\[49, 50\]](#)

Self Attention Generative Adversarial Network (SAGAN)

En las redes [Self-Attention GAN \(SAGAN\)](#) [\[51\]](#) permiten modelar dependencias de largo alcance usando técnicas de atención para tareas de generación de imágenes. Estas redes permiten generar detalles más específicos usando pistas del mapa de características, la red D puede comprobar que los rasgos detallados de las instancias generadas son coherentes entre sí.

2.3.7. La seguridad de las redes neuronales

El estado del arte de la seguridad informática es muy complejo, ya que los delincuentes informáticos adoptan nuevas técnicas y tácticas para eludir las medidas de

seguridad, esto hace que las ciber amenazas están en constante evolución, también amenaza a las tecnologías que emplean la inteligencia artificial y redes neuronales.

A continuación, se explicarán las distintas amenazas que tiene una red neuronal y como puede ser atacada o implementar defensas, posteriormente analizaremos el estado del arte de la inteligencia artificial que se emplea para mejorar la seguridad informática de los dispositivos y de los usuarios.

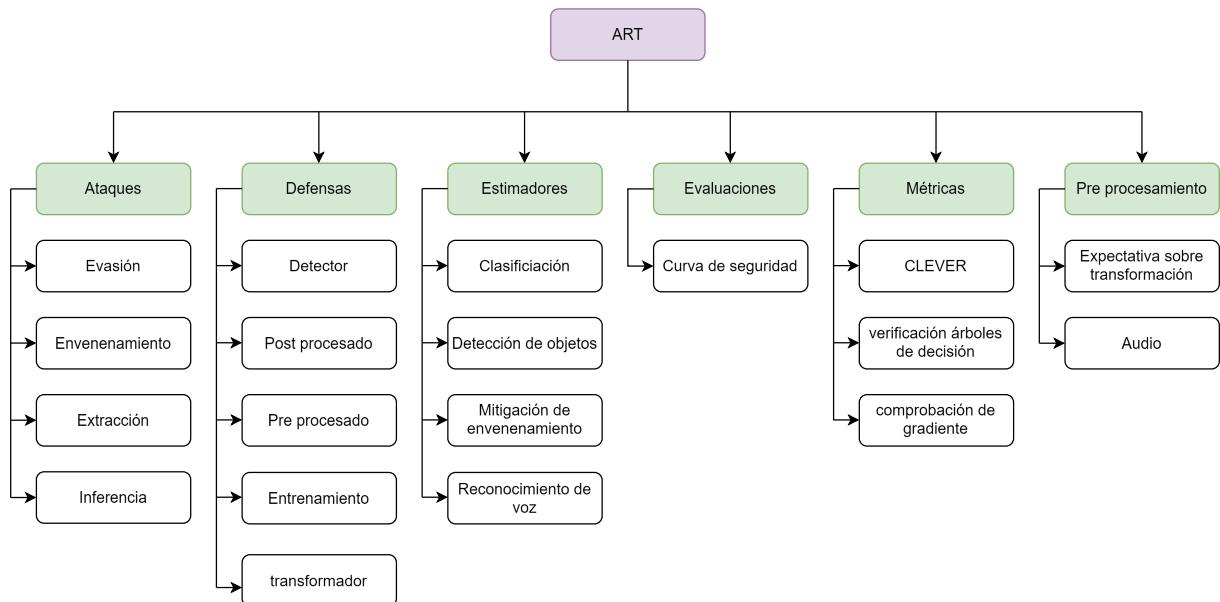


Figura 2.24: Arquitectura de ART.

Fuente: [GitHub @Trusted-AI/adversarial-robustness-toolbox](https://github.com/Trusted-AI/adversarial-robustness-toolbox)

ART [52] consta con 6 módulos específicos para ataque, defensas, estimadores, evaluaciones, métricas y preprocesamiento. Esto podemos verlo en la Figura 2.24.

2.3.8. Amenazas de *deep learning*

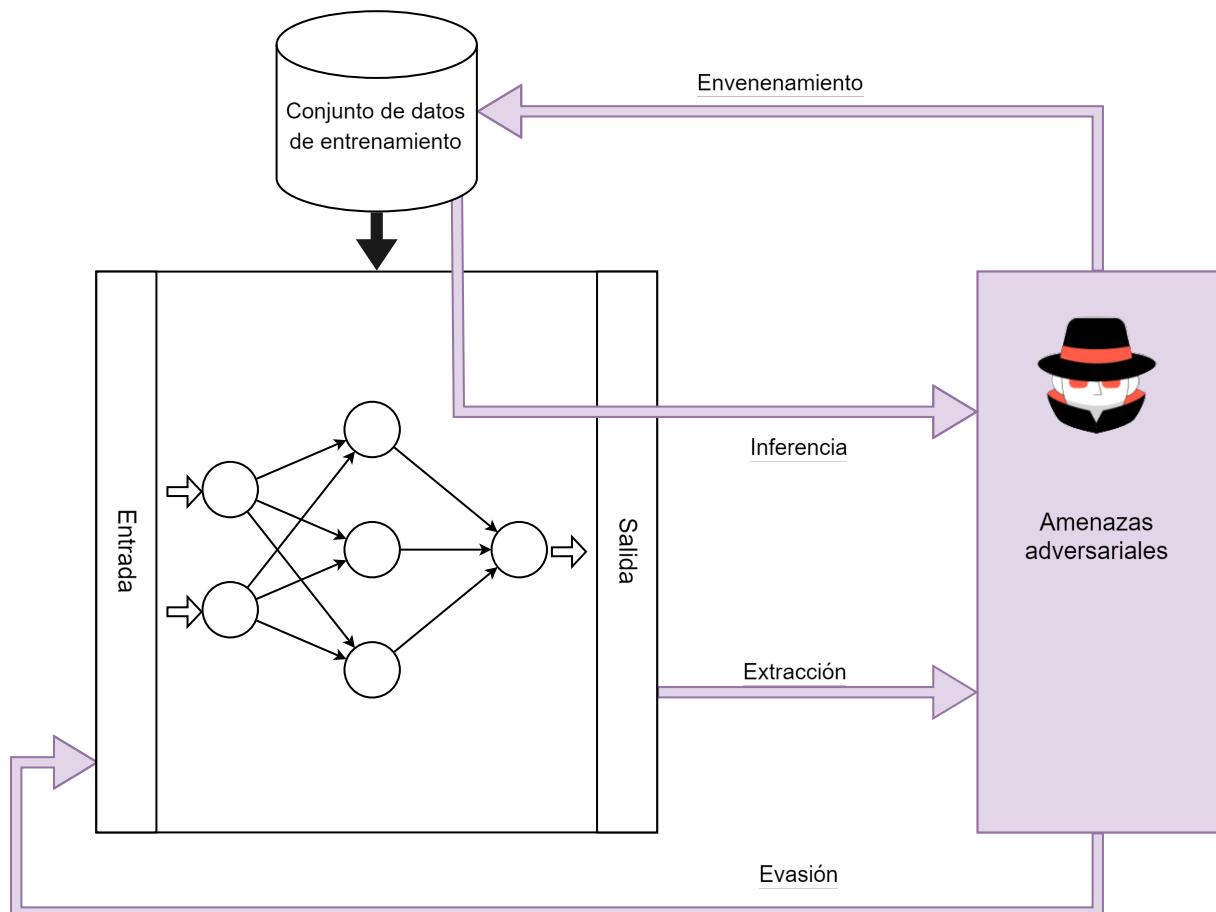


Figura 2.25: Amenazas adversariales.
Fuente: Elaboración propia.

Como podemos observar en la Figura 2.25 existen multitud de amenazas, existen cuatro formas de clasificar las amenazas, esta clasificación es en función de cómo se altera el comportamiento del modelo.

2.3.8.1. Evasión

Busca explotar las vulnerabilidades de la red neuronal para inducir errores en la capacidad de clasificar o de predecir. Se envía información con perturbaciones mínimas que alteren notablemente la respuesta del modelo.

1. **Caja blanca:** se conoce todo sobre el modelo, arquitectura, pesos, umbrales, formato de entrada de datos y salida, etc. [53]
2. **Caja negra:** se desconoce el modelo, arquitectura, pesos, umbrales, pero se conoce el formato de entrada de datos y la respuesta del modelo. [53]

2.3.8.2. Envenenamiento

Los atacantes intentan manipular los datos de entrenamiento con el objetivo de influir en el resultado del aprendizaje del modelo, esto pueden hacerlo desde distintas etapas.

1. *Label Flipping Attacks* - Ataques de cambio de etiquetas:
2. *Clean Label Data Poisoning Attack* - Ataques de envenenamiento de datos de etiquetado limpio:
3. *Backdoor Attack* - Ataques de puerta trasera: los ataques de puerta trasera, ataques que contienen un disparador, estos se pueden clasificar en tres tipos de escenarios.

2.3.8.2.1 Label Flipping Attacks

2.3.8.2.2 Clean Label Data Poisoning Attack

Feature Collision Attack

$$x_p = \underset{x}{\operatorname{argmin}} \|f(x) - f(x_t)\|_2^2 + \beta \|x - x_b\|_2^2 \quad (2.8)$$

Convex Polytope Attack and Bullseye Polytope Attack

2.3.8.2.3 Backdoor Attack

2.3.8.3. Extracción

Se busca extraer un funcionamiento interno equivalente de la red neuronal, se puede considerar un ataque de ingeniería inversa.

2.3.8.4. Inferencia

2.3.8.4.1 Inferencia por Atributos

2.3.8.4.2 Inferencia por Pertenencia

2.3.8.4.3 Inversión de modelos

2.3.8.4.4 Reconstrucción

2.3.9. Defensas contra los ataques en *deep learning*

2.3.9.1. Preprocesamiento

2.3.9.2. Post procesamiento

2.3.9.3. Entrenamiento

2.3.9.4. Transformación

2.3.9.4.1 Evasión

2.3.9.4.2 Envenenamiento

2.3.9.5. Detector

2.3.9.5.1 Evasión

2.3.9.5.2 Envenenamiento

2.3.10. Métricas en los ataques adversariales

2.3.11. Redes neuronales en *Red team* y *Blue team*

Un buen símil entre la seguridad informática clásica y la seguridad en redes neuronales son los conceptos de equipos de ataque y defensa (“red team” y “blue team”).

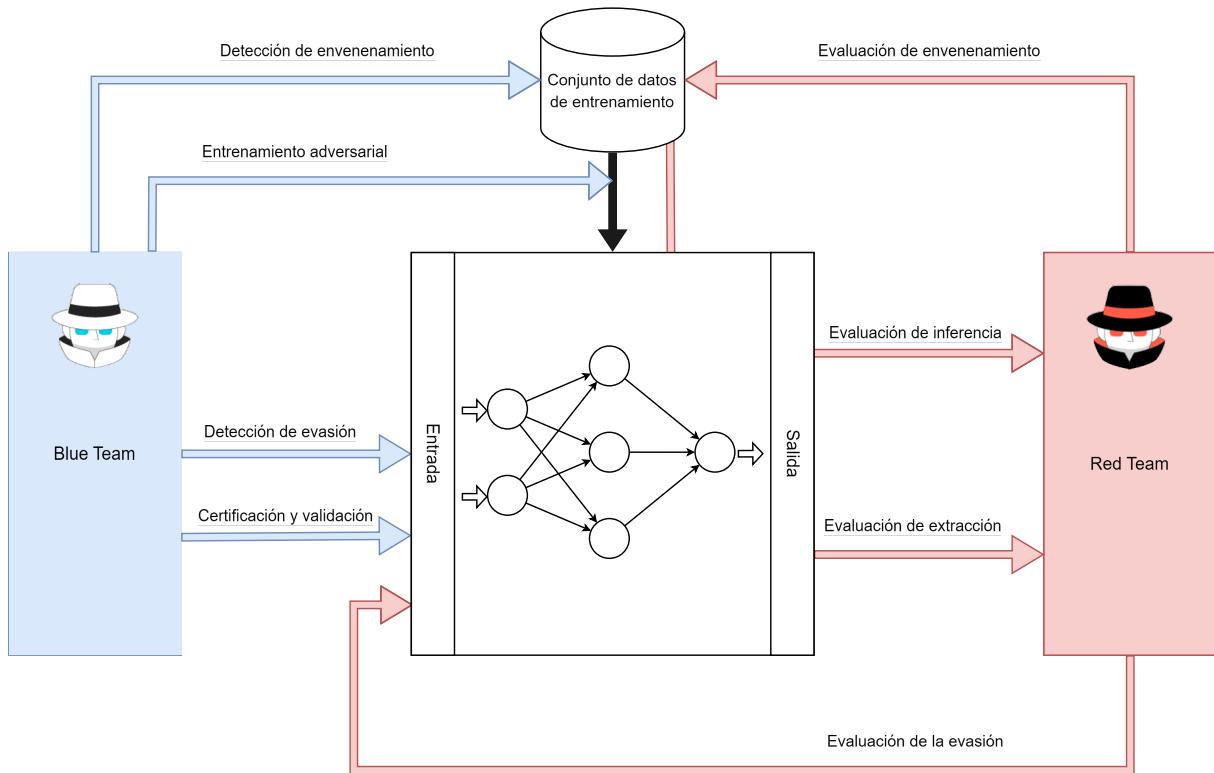


Figura 2.26: Ataques y defensas en redes neuronales.
Fuente: Elaboración propia.

La idea detrás del aprendizaje automático es la de poder predecir modelos predictivos, para ello se usan conjuntos de datos para el entrenamiento.

Los conjuntos de datos se han de procesar, ya que suelen contener mucho ruido, es decir, información muy poco valiosa, errónea o, por el contrario, una alta dimensionalidad de los datos que puede ser contraproducente por no poder reproducir esas medidas o por la poca información que aportan.

Los ataques al aprendizaje automático o profundo suelen estar dirigidos a las distintas etapas de creación y uso de un modelo.

- Alteración de los datos de entrada.
- Alteración del proceso de aprendizaje.
- Extracción de los datos de entrenamiento.
- Bloqueo del modelo.

Durante el transcurso del trabajo discutiremos los siguientes ataques y defensas previamente en la sección [La seguridad de las redes neuronales](#) que se pueden usar en los modelos.

- Envenenamiento de datos.

-
- Puerta trasera.
 - Extracción de datos
 - Ingeniería inversa.
 - Evasión.

Se puede ser categorizar la seguridad de los modelos con el modelado **STRIDE** de microsoft¹ o con la matriz **MITRE**.

Amenaza	Tipo de ataque
Corrupción de datos	Ataques de envenenamiento de datos, ataques de puerta trasera
Extracción de datos	Ataques de inferencia de membresía, ataques de ingeniería inversa
Denegación	Ataques de envenenamiento de datos

Tabla. 2.2: Amenazas STRIDE relacionadas con la seguridad de la IA

2.3.12. Las redes neuronales para la seguridad informática

2.3.13. Normativa y estándares

¹Modelado de amenazas de microsoft [Enlace](#)

Glosario

AI La inteligencia artificial (IA), también conocida por su nombre inglés, Artificial Intelligence (AI), es una tecnología que trata de realizar las tareas y tomar las decisiones empresariales de forma automática y autónoma, aprendiendo de forma continua. [54]. 7

ANN Red neuronal artificial. 27, 34

BP Método utilizado en redes neuronales para calcular el gradiente y el cálculo de los pesos, es una abreviatura de “propagación de errores hacia atrás”. 17

BP Ataque por patrones de puerta trasera. 17

cGAN . 39

CNN Red neuronal convolucional. 17, 40

CycleGAN . 39

DCGAN . 39, 40

DL Deep learning (DL), también conocido como aprendizaje profundo, es un tipo de machine learning que se estructura inspirándose en el cerebro humano y sus redes neuronales. El aprendizaje profundo procesa datos para detectar objetos, reconocer conversaciones, traducir idiomas y tomar decisiones. Al ser un tipo de machine learning, esta tecnología sirve para que la inteligencia artificial aprenda de forma continua. [55]. 7

GAN Red generativa adversarial. 37, 38

GNN Es una clase de redes neuronales especializada en el procesamiento de datos que se puedan representar como gráficos. 17

KDD Es el proceso utilizado para extraer de forma eficiente y automática **información útil** a partir de grandes volúmenes de datos. 11, 20

LSGAN . 40

LTU La unidad de umbral lineal es una neurona artificial muy simple cuya salida es la sumatoria de la entrada total umbralizada. Es decir, una LTU con umbral T calcula la suma ponderada de sus entradas y, a continuación, emite **0** si esta suma

es inferior a T y 1 si la suma es superior a T. Las LTU constituyen la base de los perceptrones. [56]. 27, 28

MITRE La abreviatura de (Tácticas, Técnicas y Conocimiento Amplio de Enemigos) es un marco de trabajo para la evaluación de la seguridad en las organizaciones. [Enlace](#). 8, 46

ML El machine learning es la tecnología que permite que un sistema aprenda de forma continua. El sistema recibe un input, un humano reacciona y, así, la próxima vez que el sistema reciba ese input, sabrá cómo actuar sin necesidad de acudir al humano. [57]. 7

MLP El Perceptrón multicapa es una red neuronal artificial formada por capas local o totalmente conectadas. 27

Pix2Pîx . 39

ProGAN . 40

SAGAN . 40

SGAN . 38

SRGAN . 40

STRIDE Son las siglas de las amenazas de *Spoofing, Tampering, Repudiation, Information, Denial, Elevation* que violan las propiedades de **Autenticidad, Integridad, No Repudio, Información, Disponibilidad y Autorización** respectivamente. 46

StyleGAN . 19

VAE Un codificador automático variacional es un tipo de modelo generativo basado en probabilidad.. 38

Siglas

ANN Artificial Neural Network. [12](#)

ART Adaptive Resonance Theory. [17](#)

DBN Deep Belief Networks. [18](#)

FNN Feedforward Neural Network. [17](#), [18](#)

GAN Generative Adversarial Network. [17](#), [19](#)

LSTM Long Short-Term Memory Network. [18](#)

NN Neural Network. [14](#), [18](#)

RNN Recurrent Neural Network. [18](#)

Bibliografía

- [1] Contenido y Tecnologías (Comisión Europea) Comisión Europea, Dirección General de Redes de Comunicación. Libro blanco sobre la inteligencia artificial - un enfoque europeo orientado a la excelencia y la confianza. Technical report, Comisión Europea, 2020.
- [2] Pytorch. <https://github.com/pytorch/pytorch>, 2024.
- [3] Gottfried Wilhelm Leibniz. *The early mathematical manuscripts of Leibniz*. Courier Corporation, 2012.
- [4] Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251(254): 10, 2012.
- [5] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.
- [6] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [7] Eric B Baum. On the capabilities of multilayer perceptrons. *Journal of complexity*, 4(3):193–215, 1988.
- [8] Walter J Karplus. 1967 index ieee transactions on electronic computers vol. ec-16. *IEEE Transactions on Electronic Computers*, EC-16(6):913–932, 1967.
- [9] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.
- [10] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Cornell Aeronautical Laboratory. Report no. VG-1196-G-8. Spartan Books, 1962. URL <https://books.google.es/books?id=7FhRAAAAMAAJ>.
- [11] P J Werbos. Applications of advances in nonlinear sensitivity analysis, Jan 1982.
- [12] Yann LeCun. A learning scheme for asymmetric threshold networks. *Proceedings of COGNITIVA*, 85(537):599–604, 1985.

-
- [13] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985. ISSN 0364-0213. doi: [https://doi.org/10.1016/S0364-0213\(85\)80012-4](https://doi.org/10.1016/S0364-0213(85)80012-4). URL <https://www.sciencedirect.com/science/article/pii/S0364021385800124>.
 - [14] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.
 - [15] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
 - [16] Jürgen Schmidhuber. Habilitation thesis: System modeling and optimization. *ff demonstrates credit assignment across the equivalent of*, 1:150, 1993.
 - [17] Jürgen Schmidhuber and AI Blog. Unsupervised neural networks fight in a minimax game. *Juergen Schmidhuber's AI Blog*, 1990s.
 - [18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997. URL <https://api.semanticscholar.org/CorpusID:1915014>.
 - [19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
 - [20] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
 - [21] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
 - [22] Jean-Arcady Meyer and Stewart W. Wilson. *A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers*, pages 222–227. The MIT Press, 1991.
 - [23] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.
 - [24] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.
 - [25] J. C. Cuevas-Tello. Apuntes de redes neuronales artificiales, 2018.
 - [26] Bernard Widrow. An adaptive “adaline” neuron using chemical “memistors”, 1553–1552, 1960.
-

- [27] K Fukushima. Neural network model for pattern recognition mechanism not affected by position deviation-neocognitron. *Trans.(A) IECE, Japan*, pages 658–665, 1979.
 - [28] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
 - [29] J urgen Schmidhuber. Learning factorial codes by predictability minimization (compact version of tr cu-cs-565-91). *Juergen Schmidhuber's AI Blog*, 1991.
 - [30] AWS. ¿qué es la ciencia de datos? <https://aws.amazon.com/es/what-is/data-science/>, 2023. [Online; accessed 31-January-2024].
 - [31] Lior Rokach and Oded Maimon. *Data mining and knowledge discovery handbook*. Springer New York, 2010.
 - [32] Oded Maimon and Lior Rokach. *Data mining and knowledge discovery handbook*, volume 2. Springer, 2005.
 - [33] IBM. ¿qué es el deep learning? <https://www.ibm.com/es-es/topics/deep-learning>, 2023. [Online; accessed 1-February-2024].
 - [34] A. Géron. *Neural Networks and Deep Learning*. O'Reilly, 2018. URL <https://books.google.es/books?id=5pm6tQEACAAJ>.
 - [35] IBM. ¿qué es el descenso de gradiente? <https://www.ibm.com/mx-es/topics/gradient-descent>, 2023. [Online; accessed 10-February-2024].
 - [36] Jordi de la Torre. Redes generativas adversarias (gan) fundamentos teóricos y aplicaciones. *arXiv preprint arXiv:2302.09346*, 2023.
 - [37] Augustus Odena. Semi-supervised learning with generative adversarial networks, 2016.
 - [38] Yufeng Zheng, Yunkai Zhang, and Zeyu Zheng. Continuous conditional generative adversarial networks (cgan) with generator regularization, 2021.
 - [39] Xiaomin Li, Anne Hee Hiong Ngu, and Vangelis Metsis. Tts-cgan: A transformer time-series conditional gan for biosignal data augmentation, 2022.
 - [40] Saul Dobilas Towards Data Science. cgan: Conditional generative adversarial network — how to gain control over gan outputs. cGAN: Conditional Generative Adversarial Network - How to Gain Control Over GAN Outputs, 2022.

-
- [41] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.
 - [42] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.
 - [43] Jairo Viola, YangQuan Chen, and Jing Wang. Faultface: Deep convolutional generative adversarial network (dcgan) based ball-bearing failure detection method, 2020.
 - [44] Süleyman Aslan, Uğur Gündükay, B. Uğur Töreyin, and A. Enis Çetin. Deep convolutional generative adversarial networks based flame detection in video, 2019.
 - [45] J. D. Curtó, I. C. Zarza, Fernando de la Torre, Irwin King, and Michael R. Lyu. High-resolution deep convolutional generative adversarial networks, 2020.
 - [46] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2017.
 - [47] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.
 - [48] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017.
 - [49] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.
 - [50] Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis, 2023.
 - [51] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2018.
 - [52] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069, 2018. URL <https://arxiv.org/pdf/1807.01069.pdf>.
 - [53] Will Schroeder. Learning machine learning part 3: Attacking black box models. <https://posts.specterops.io/learning-machine-learning-part-3-attacking-black-box-models-3efffc256909>, 2022. [Online; accessed 30-January-2024].
-

-
- [54] ticportal. Inteligencia artificial para software empresarial. <https://www.ticportal.es/glosario-tic/inteligencia-artificial-software-empresarial>, 2022. [Online; accessed 30-January-2024].
- [55] ticportal. Deep learning: ¿se pueden programar las máquinas para pensar como humanos? <https://www.ticportal.es/glosario-tic/deep-learning-dl>, 2023. [Online; accessed 30-January-2024].
- [56] Bill Wilson. The machine learning dictionary. <https://www.cse.unsw.edu.au/~billw/mldict.html>, 2012.
- [57] ticportal. Machine learning: ¿cómo aprenden las máquinas a trabajar por su cuenta? <https://www.ticportal.es/glosario-tic/machine-learning>, 2022. [Online; accessed 30-January-2024].
- [58] Francisco Charte. A comprehensive and didactic review on multilabel learning software tools. *IEEE Access*, 8:50330–50354, 2020. doi: 10.1109/ACCESS.2020.2979787.
- [59] Francisco Herrera, Francisco Charte, Antonio J Rivera, and María J del Jesus. *Multilabel Classification: Problem Analysis, Metrics and Techniques*. Springer, 2016. ISBN 978-3-319-41110-1. doi: 10.1007/978-3-319-41111-8.
- [60] Francisco Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. A first approach to deal with imbalance in multi-label datasets. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 150–160. Springer, 2013. doi: 10.1007/978-3-642-40846-5_16.
- [61] Francisco Charte. Cómo analizar la distribución de los datos con R, 2019. URL <https://fcharte.com/tutoriales/20170114-DistribucionDatosR/>. comprobado en 2020-09-30.
- [62] SE Fahlman. Proceedings of the 1988 connectionist models summer school. In *Faster-learning variations on back-propagation: An empirical study*, 1988.
- [63] Juergen Schmidhuber. Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991), 2020.
- [64] El señor de las gan: el hombre que dio imaginación a las máquinas, 2014. URL <https://www.technologyreview.es/s/10016/el-señor-de-las-gan-el-hombre-que-dio-imaginacion-las-maquinias>.
-

-
- [65] J. M. Child. The manuscripts of leibniz on his discovery of the differential calculus. part ii (continued). *The Monist*, 27(3):411–454, 1917. ISSN 00269662. URL <http://www.jstor.org/stable/27900650>.
 - [66] Will Schroeder. Learning machine learning part 1: Introduction and revoke-obfuscation. <https://posts.specterops.io/learning-machine-learning-part-1-introduction-and-revoke-obfuscation-c73033184f0>, 2022. [Online; accessed 30-January-2024].
 - [67] Will Schroeder. Learning machine learning part 2: Attacking white box models. <https://posts.specterops.io/learning-machine-learning-part-2-attacking-white-box-models-1a10bbb4a2ae>, 2022. [Online; accessed 30-January-2024].
 - [68] Jing Lin, Long Dang, Mohamed Rahouti, and Kaiqi Xiong. Ml attack models: Adversarial attacks and data poisoning attacks, 2021.
 - [69] Bill Wilson. The ai dictionary. <https://www.cse.unsw.edu.au/~billw/aidict.html>, 2012.
 - [70] Mahmoud Afifi. 11k hands: gender recognition and biometric identification using a large dataset of hand images. *Multimedia Tools and Applications*, 2019. doi: 10.1007/s11042-019-7424-8. URL <https://doi.org/10.1007/s11042-019-7424-8>.
 - [71] Wenxiu Diao, Feng Zhang, Jiande Sun, Yinghui Xing, Kai Zhang, and Lorenzo Bruzzone. Zergan: Zero-reference gan for fusion of multispectral and panchromatic images. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8195–8209, 2023. doi: 10.1109/TNNLS.2021.3137373.
 - [72] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962. doi: 10.1109/TIT.1962.1057692.
 - [73] Rafael M. Luque-Baena, David Elizondo, Ezequiel López-Rubio, Esteban J. Palomo, and Tim Watson. Assessment of geometric features for individual identification and verification in biometric hand systems. *Expert Systems with Applications*, 40(9):3580–3594, 2013. ISSN 0957-4174. doi: <https://doi.org/10.1016/j.eswa.2012.12.065>. URL <https://www.sciencedirect.com/science/article/pii/S0957417412013061>.
 - [74] Redacción KeepCoding. Visualización de activaciones y filtros en red convolucional. <https://keepcoding.io/>, 2022. [Online; accessed 20-February-2024].
-