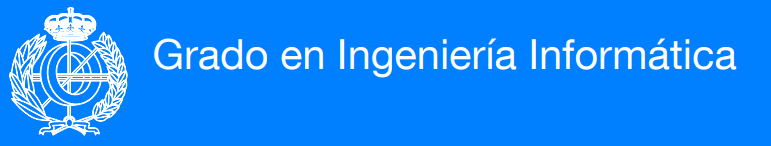
A blue sign with white text

Description automatically generated

Reconocimiento visual de aves con Deep Learning





Trabajo Fin de Grado

Autor: Juan García Martínez

Tutor/es:

José García Rodríguez

Esther Sebastián González

02/2024

2024

Contenido

[Palabras clave 4](#_Toc169162149)

[1. Introducción 4](#_Toc169162150)

[1.1 Justificación 4](#_Toc169162151)

[1.2 Estructura del TFG 4](#_Toc169162152)

[2. Objetivos 5](#_Toc169162153)

[3. Marco teórico 5](#_Toc169162154)

[3.1 Introducción a la inteligencia artificial 5](#_Toc169162155)

[3.2 Machine Learning 7](#_Toc169162156)

[3.3 Deep Learning 8](#_Toc169162157)

[3.4 Transfer Learning 9](#_Toc169162158)

[3.4.1 Transfer Learning en Deep Learning 9](#_Toc169162159)

[3.4.2 Transfer Learning en Machine Learning 10](#_Toc169162160)

[3.5 Redes Neuronales Artificiales 10](#_Toc169162161)

[3.5.1 Redes Neuronales Convolucionales 10](#_Toc169162162)

[3.5.2 Redes Neuronales Recurrentes 14](#_Toc169162163)

[3.5.3 Redes Generativas Adversarias 16](#_Toc169162164)

[4. Estado del arte 17](#_Toc169162165)

[4.1 Cornell Lab of Ornithology 17](#_Toc169162166)

[4.1.1 eBird 18](#_Toc169162167)

[4.1.2 Merlin Bird ID 19](#_Toc169162168)

[4.2 iNaturalist 21](#_Toc169162169)

[5. Metodología 22](#_Toc169162170)

[5.1 Hardware 22](#_Toc169162171)

[5.2 Software 23](#_Toc169162172)

[5.2.1 Python 23](#_Toc169162173)

[5.2.2 Google Colab con Python 23](#_Toc169162174)

[5.2.3 Anaconda Distribution con Python 24](#_Toc169162175)

[5.2.4 CUDA ToolKit y cuDNN 25](#_Toc169162176)

[5.2.5 Librerías 26](#_Toc169162177)

[5.3 Dataset 30](#_Toc169162178)

[6. Desarrollo 30](#_Toc169162179)

[6.1 Obtención de datos 30](#_Toc169162180)

[6.2 Análisis de datos 30](#_Toc169162181)

[6.3 Modelos 30](#_Toc169162182)

[6.4 Experimentación 30](#_Toc169162183)

[7. Resultados 31](#_Toc169162184)

[8. Conclusión 31](#_Toc169162185)

[Bibliografía 31](#_Toc169162186)

[Referencias 34](#_Toc169162187)

[Lista de Acrónimos y Abreviaturas 35](#_Toc169162188)

# Palabras clave

Inteligencia artificial, Machine Learning, Deep Learning, Tranfer Learning, Redes neuronales Convolucionales, dataset, aves, arquitectura.

# 1. Introducción

En este proyecto vamos a estudiar el uso de la Inteligencia Artificial y las tecnologías para la detección de imágenes automáticamente, concretamente ente imágenes de aves.

En este apartado veremos la motivación que hizo interesarme por esta propuesta y los objetivos planteados para completarla.

## 1.1 Justificación

La elección de esta propuesta es mayoritariamente una motivación personal ya que me fascinan las aves, conocer su hábitat, su comportamiento y la conservación de las especies.

Considero que los parques naturales representan un escenario fundamental en la gestión para la sostenibilidad de la biodiversidad a largo plazo. Son espacios donde se concentra la protección de ecosistemas frágiles, la investigación científica y la educación ambiental, desempeñando un papel fundamental en la preservación de la diversidad biológica y en el desarrollo de prácticas sostenibles.

La decisión para contemplar este tema en el Trabajo Fin de Grado no se limita a satisfacer mis intereses personales, sino que también para ampliar mis conocimientos sobre un área que me interesa y en la que todavía tengo mucho que aprender.

## 1.2 Estructura del TFG

A continuación, se detallará la estructura del TFG y las tareas que se deben de realizar para su desarrollo:

* Objetivos: Se definirán los objetivos que se pretende alcanzar mediante el desarrollo del TFG
* Marco teórico: Se explicarán las bases y el contexto de la rama a la que pertenece el TFG, como la Inteligencia Artificial (IA), Machine Learning (ML), Deep Learning (DL), Redes Neuronales Artificiales (ANN) y las Arquitecturas.
* **Estado del arte:** Se mostrarán las tecnologías empleadas para el desarrollo de este TFG.
* **Metodología:** Se mostrarán las herramientas empleadas en este TFG.
* **Resultados:** Se analizarán los resultados con las diferentes pruebas sobre sus arquitecturas y parámetros.
* **Conclusión:** Se realizará un análisis de los resultados obtenidos para extraer conclusiones y determinar si se han alcanzado los objetivos propuestos.

# 2. Objetivos

El objetivo principal de este TFG es realizar un sistema capaz de reconocer y clasificar visualmente aves utilizando técnicas de Deep Learning. Este sistema podrá identificar automáticamente las aves mediante imágenes y para ello se empleará un conjunto de imágenes.

Además, se investigarán los procesos, arquitecturas, modelos y tecnologías más empleadas por la comunidad científica para el reconocimiento visual de aves. El objetivo final es desarrollar un modelo ligero y eficiente que puedan ser implementados en dispositivos como Raspberri Pi o Jetson Nano, permitiendo realizar un seguimiento continuo y automatizado de las aves y su comportamiento en los humedales de la provincia de Alicante.

# 3. Marco teórico

En este capítulo se introducirán las bases teóricas para el posterior entendimiento del proyecto, incluyendo las tecnologías, arquitecturas y herramientas de la Inteligencia artificial que se emplean hoy en día para el reconocimiento visual de imágenes, concretamente de aves.

## 3.1 Introducción a la inteligencia artificial

La **inteligencia artificial (IA)** es un campo de la informática donde su principal objetivo es la creación de sistemas que puedan realizar labores que normalmente requieren de la inteligencia humana, como el razonamiento, percepción y el aprendizaje.

Los orígenes de la inteligencia artificial se remontan a la década de 1950, cuando contempla la posibilidad de crear maquinas que pudiera simular la inteligencia humana.

En 1956 se realizó la Conferencia de Dartmouth College, organizada por Marvin Minsky, John McCarthy y Claude Shanon, en la cual se reunió a todos los que trabajaban en el campo de la inteligencia artificial.



Figura 2.1: Conferencia de Dartmouth College 1956. De derecha a izquierda: John McCarthy, Oliver Selfridge, Marvin Minsky, Trenchard More y Ray Solomonoff (<https://infinitomaizum.wordpress.com/2009/11/05/em-1956-nasce-a-inteligencia-artificial/>)

En esta conferencia apareció el termino de inteligencia artificial por primera vez y se prefijó que en 25 años los ordenadores serían capaces de hacer todo el trabajo de los humanos, algo que hoy en día podemos comprobar que fueron optimistas.

Dos años después, en 1958 John McCarthy inventa el lenguaje de programación LISP [1] que se convierte en uno de los principales lenguajes de programación para IA.



Figura 2.2: Libro LISP 1.5 Programmer’s Manual creado por John McCarthy (<https://onih.pastperfectonline.com/webobject/7D1DFC2C-E98B-4491-9EEA-891675764224>)

Tras las prometedoras declaraciones y avances iniciales en la inteligencia artificial, surgió “el primer invierno de la Inteligencia Artificial” que fue un periodo, generalmente ubicado en la década de 1970, en el cual hubo grandes fracasos por las promesas poco realistas por parte de los desarrolladores y las altas expectativas que tenían los usuarios finales.

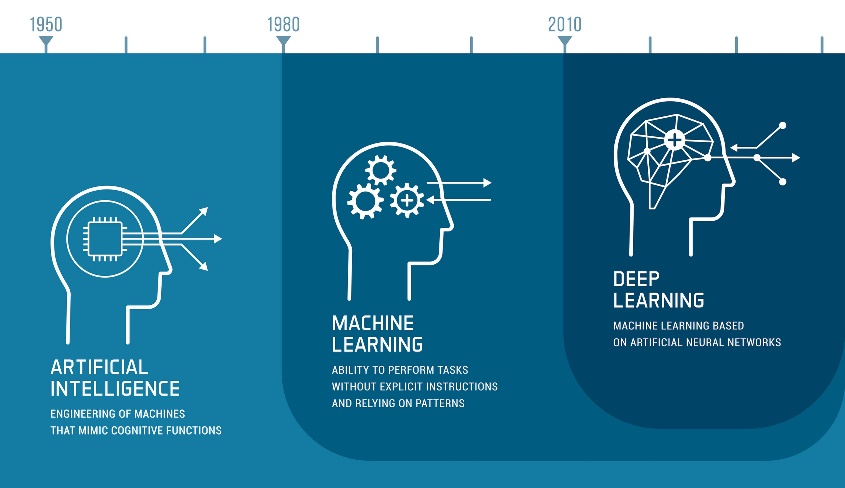
Principalmente, este estancamiento surgió por las limitaciones de esa época y los programadores e investigadores se encontraron con desafíos que no tenían una resolución ligera, como la lentitud de los procesos y la baja capacidad de memoria. Esto sentó precedente y empezó el declive del interés, inversión y la investigación de esta tecnología.

Sin embargo, el primer invierno de la IA llegó a su fin con la introducción de los sistemas expertos (SE) [2]. Estos sistemas, que comenzaron a ser adoptados por diversas organizaciones en todo el mundo, marcando un punto de inflexión que reavivó el interés y la inversión en la inteligencia artificial.

En las décadas posteriores, los investigadores tomaron nuevos enfoques respecto a la IA intentado abordar nuevos ámbitos como el comportamiento e inteligencia humana, la robótica, Procesamiento de Lenguaje Natural (PLN), Aprendizaje Automático Temprano y lógica y razonamiento.

Actualmente, gracias a los avances de las tecnológicos, como el aumento de potencia del cómputo y el almacenamiento, se han desarrollado nuevas técnicas y algoritmos que tienen la capacidad de procesar información derivada de datos y tomar decisiones para lograr un objetivo dado, dicho de otra manera, que puedan pensar y actuar como humanos e incluso superarlo en algunos ámbitos.

En este proyecto nos centraremos en las ramas de la Inteligencia Artificial, Machine Learning y Deep Learning.

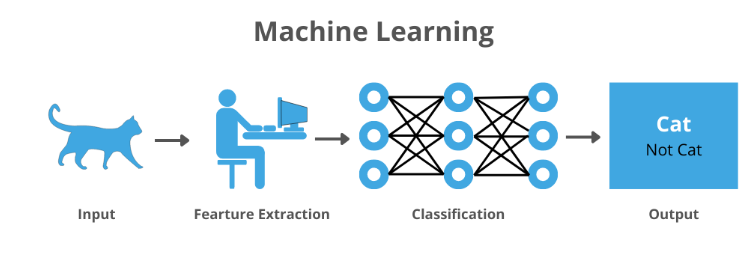


**Figura 3.1:** Inteligencia Artificial, Machine Learning y Deep Learning (<https://computerhoy.com/tecnologia/deep-learning-considera-revolucion-ia-1241180>)

3.2 Machine Learning

Machine Learning (ML), también conocido como aprendizaje automático, es una subdisciplina de la IA que permite aprender de los datos utilizando algoritmos para identificar patrones, realizar predicciones y mejorar el rendimiento sin estar programadas para ello.

El proceso de ML se constituye principalmente por 3 componentes: el modelo, la entrada y la salida de datos. El modelo es un algoritmo que aprende patrones a partir de los datos, la entrada de datos es el conjunto de datos que se emplea para entrenar el modelo y los datos de salida son los resultados generados por la predicción del modelo tras el procesamiento de los datos de entrada.



**Figura 3.2:** Esquema del proceso de Machine Learning (<https://intersog.co.il/blog/ais-powerful-duo-difference-between-machine-learning-and-deep-learning/>)

Existen diversos tipos de ML, dependiendo de la forma en la que aprenden: El aprendizaje supervisado, aprendizaje no supervisado y aprendizaje de refuerzo.

El aprendizaje supervisado entrena con una base de datos con información correctamente etiquetada que se utiliza posteriormente para predecir nuevos datos sin etiquetar. El aprendizaje no supervisado entrena con datos sin etiquetar detectando patrones o estructuras ocultas en los datos. Por último, el aprendizaje reforzado forma parte de lo que se conoce como Deep Learning y tiene la finalidad de construir modelos que aumenten el rendimiento y realicen predicciones tomando como base los resultados, ya sean recompensas o castigos, de cada iteración realizada en el modelo.

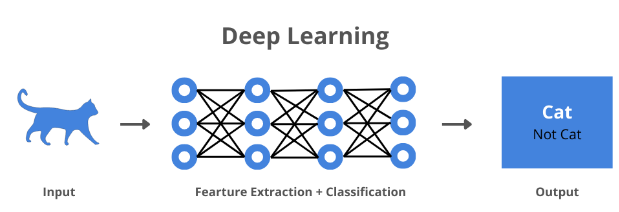
ML se emplea en diversos ámbitos, desde la detección de spam, completar texto y recomendación de contenido hasta servicios en línea, predicción del clima y diagnósticos médicos.

Existen gran cantidad y diversidad de algoritmos de ML, algunos de ellos incluyen Redes Neuronales, los Árboles de decisión (supervisado), K means (No supervisado) y Deep Q-Networks (Reforzado).

## 3.3 Deep Learning

Deep Learning (DP) es una subrama del ML que se enfoca en el uso de ANN para modelar y aprender parones complejos en gran cantidad de datos de entrada, que a diferencia de ML emplea capas de neuronas artificiales extrayendo automáticamente características de alto nivel directamente de los datos.

A las capas de entrada y salida se les denominan capas visibles, en la capa de entrada es ingiriere los datos para el procesado y la capa de salida es donde se realiza la predicción final. Entre estas capas se encuentran las neuronas que se denominan las capas ocultas, de ahí el término “Deep” (profundo), donde capa tras capa se aprenden características jerárquicamente, donde las capas iniciales aprenden características simples y las más profundas características más abstractas y complejas dando lugar a dando lugar a representaciones complejas que mejoran la precisión de las predicciones en la capa de salida.



**Figura 3.2:** Esquema del proceso de Deep Learning (<https://intersog.co.il/blog/ais-powerful-duo-difference-between-machine-learning-and-deep-learning/>)

Respecto al uso del DL, es muy útil en áreas como VC, NLP, ciencias de la salud y reconocimiento de voz y audio, gracias a su capacidad para manejar y extraer información de grandes volúmenes de datos. Con el avance y mejoras sustanciales en los algoritmos, el DL se emplea en diversas áreas desde el marketing personalizado hasta la conducción autónoma y el diagnóstico médico.

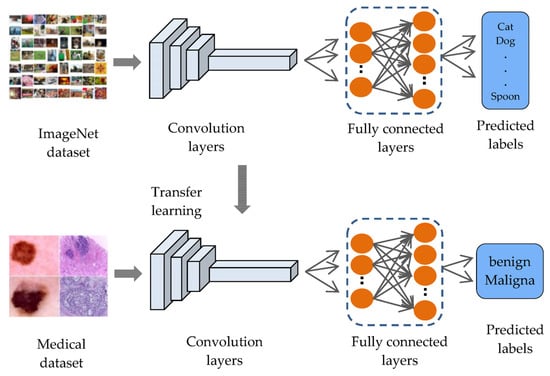
En resumen, actualmente el DP es una herramienta de vital importancia gracias a la capacidad de trabajar de manera autónoma. Sin embargo, presenta ciertas limitaciones como los altos requerimientos computacionales, necesidad de grandes conjuntos de datos y el riesgo de sobreajuste, más conocido como overfitting.

## 3.4 Transfer Learning

El Transfer Learning es un método de ML que aprovecha el entrenamiento de una red adquirido en una terea determinada para emplear como punto de partida en el desarrollo de otro modelo con otra terea diferente pero relacionada. En el contexto de DL. su uso es muy popular debido a su efectividad y capacidad de mejorar el rendimiento de los modelos y acelerar los procesos de entrenamiento.

### 3.4.1 Transfer Learning en Deep Learning

En DL, Transfer Learning se emplea principalmente en CNN preentrenadas. Este uso es especialmente útil cuando se parte de un conjunto de datos limitado para la nueva tarea, ya que el modelo empleado como punto de partida es entrenado con un conjunto de datos grandes y diversos.



**Figura 3.3:** Esquema Transfer Learning en DL (<https://www.mdpi.com/1424-8220/23/2/570>)

Algunas aplicaciones de Transfer Learning dentro de DL son las siguientes:

* **Modelo Preentrenado:** Emplear una CNN preentrenada con un conjunto de datos grande.
* **Fine-Tuning:** Reemplazar y entrenar las capas finales del modelo con el nuevo conjunto de datos específico para la nueva tarea.
* **Congelación de capas:** Se congela las primeras capas del modelo ya que aprenden características muy genéricas, mientas que las ultimas se ajustan a la tarea específica.

### 3.4.2 Transfer Learning en Machine Learning

Aunque Transfer Learning es más común emplearlo con DL gracias a la capacidad de aprendizaje en modelos complejos, también se emplea en ML mejorando modelos de NLP ajustándose a otras tareas de NLP.

Algunas aplicaciones de Transfer Learning en ML son las siguientes:

* NLP: En modelos preentrenados de grandes textos, pudiendo ser entrenados para tareas más específicas como la clasificación de texto y la traducción automática.
* Reconocimiento de Voz y Audio: Empleado en modelos con grandes datos de audios para ajustarse a tareas específicas como la detección de sonidos específicos o reconocimiento de voz en distintos idiomas.

## 3.5 Redes Neuronales Artificiales

Las redes neuronales artificiales (ANN) son modelos computacionales que permite dotar a los sistemas de la capacidad de aprender, interpretar y predecir de una manera muy similar a como lo hace el cerebro humano.

Las ANN están formadas por neuronas artificiales conectadas entre sí, las cuales reciben información de los datos de entrada o de otras neuronas, de manera muy similar a las neuronas del cerebro, que reciben impulsos nerviosos. Estas ANN generan un valor de salida que se transmite a otras neuronas o a la salida final de la red. En los siguientes apartados se introducirán los tipos más comunes de ANN, cada uno de ellos diseñados para resolver distintos tipos de desafíos.

### 3.5.1 Redes Neuronales Convolucionales

En este subapartado se profundizará sobre una familia comúnmente empleada en el procesamiento de imágenes y reconocimiento visual.

Las CNN son una subcategoría de las ANN implementadas para el procesamiento y clasificación de imágenes, es decir, para aplicaciones de VC. La estructura y procesamiento están inspiradas en el sistema visual de los animales.

La arquitectura de una CNN enfocadas a la clasificación de imágenes es prácticamente igual, aplicando filtrado de imágenes, extracciones de características de cada imagen y reducción de parámetros.

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

**Figura 3.4:** Esquema de una arquitectura CNN (<https://apiumhub.com/es/caso-de-uso-deep-learning/>)

En la figura 3.4 se observa las diferentes capas fundamentales. Cada capa realiza un propósito concreto:

* La capa convolucional es la capa encargada de analizar los datos de entrada detectando patrones, texturas, bordes y otras características mediante un filtro *kernel* elegido y que recorrerá todos los píxeles de la imagen y se obtendrá una nueva matriz de salida.

Se realiza entre una matriz bidimensional de entrada y otra llamada filtro, máscara o *kernel* que se desliza por la matriz bidimensional realizando una operación matemática en cada sección de esta y almacenando el resultado en otra matriz. Esta operación realizada, consiste en una suma de los resultados de la multiplicación de los valores de la matriz filtro por los valores correspondientes de la matriz bidimensional. Esta operación se realizará iterativamente hasta desplazar el filtro por toda la matriz de entrada y completar a matriz de salida.

La Figura 3.5 ilustra la operación de convolución de una matriz de entrada 7 x 7 con un filtro de tamaño 3 x 3.

Imagen que contiene teclado

Descripción generada automáticamente

**Figura 3.5:** Operación de convolución (<https://www.diegocalvo.es/red-neuronal-convolucional/>)

* La capa de activación y sesgo implementa una función de activación transformando cada valor en la salida de la convolución. Si el valor de salida es negativo, lo cambia a cero, si es positivo lo deja igual.

Esta capa juega un papel importante ya que sin ellas no se podría modelar las complejidades de los datos ya que la red sería una combinación lineal de entradas.

Las funciones más comunes de activación son las siguientes:

**Sigmoid**

La función sigmoid es una función matemática con forma de S donde su salida está definida en el rango de 0 y 1, siendo muy útil en problemas de clasificación binaria y frecuentemente empleada en la última capa de la red neuronal.

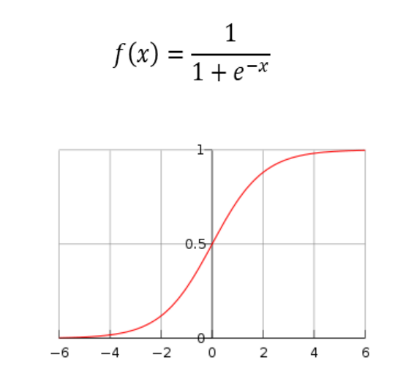


Figura 3.6: Función sigmoid.

**Softmax**

La función softmax, es conocida como función exponencial normalizada siendo muy empleada en la clasificación multiclase. Esta función opera sobre un vector convirtiéndolo en una distribución de probabilidades cuya suma de todo el vector sería uno. Este conjunto de probabilidades se podría definir como la cantidad de confianza para cada clase, proporcionando información sobre la certeza del modelo respecto a sus previsiones.

Texto

Descripción generada automáticamente con confianza mediaGráfico

Descripción generada automáticamente

Figura 3.7: Función de activación softmax.

Esta función se usa principalmente en la última capa de las rede neuronales para tareas de clasificación con múltiples clases.

**Relu**

ReLU es una de las funciones de activación más empleadas dentro de las DNN. Para las entradas mayores que cero, actúa como una función lineal con un gradiente de 1, es decir, no modifica las entradas positivas y evita a mitigar el desvanecimiento de gradiente (pérdida de parámetros en la red neuronal).

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Figura 3.8: Función de activación ReLU.

Como ReLU produce cero para todas las entradas negativas, realiza activaciones de neuronas de manera dispersa y en cualquier momento, activando solo un subconjunto de neuronas, lo que hace que computacionalmente sea más eficiente.

Esta función es utilizada principalmente en las capas ocultas de la ANN.

**Tahn**

Esta función es similar a sigmoid, pero centrada en cero produciendo valores en el rango de -1 hasta 1, pudiendo tratar los valores negativos con mejor eficacia que sigmoid. Esta permite aprendizaje y convergencias más rápidas durante el entrenamiento siendo más resistentes al desvanecimiento de gradiente que la función sigmoid.

A pesar de estas ventajas, la función Tanh también sufre el problema de desvanecimiento de gradiente. Esto puede ralentizar drásticamente el proceso de entrenamiento y provocar malas propiedades de convergencia.

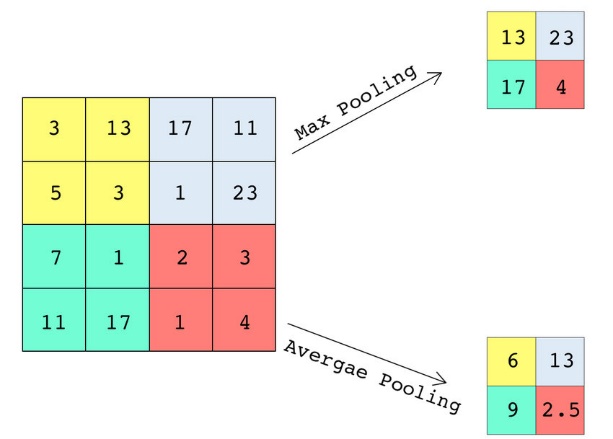
Diagrama

Descripción generada automáticamente con confianza media

Figura 3.9: Función de activación Tahn.

Tahn es empleada en las capas ocultas de las redes neuronales.

* La capa pooling reduce las dimensiones de la matriz de las características detectadas en las capas convolucionales, perseverando así las características más esenciales. Consiste en dividir la matriz de entrada en regiones no solapadas, desplazando el filtro de tamaño p1xp2 sobre la matriz de entrada y aplicar una función de agregación sobre cada una de las regiones. Esto genera una red más robusta y disminuye la cantidad de parámetros y el coste computacional. Existen varios tipos de funciones de agregación en pooling:
  + **Avg-Pooling:** Calcula la media de los valores en cada región del mapa de características.
  + **Min-Pooling:** Calcula el valor mínimo en cada región del mapa de características.
  + **Max-pooling:** Calcula el valor máximo en cada región del mapa de características, siendo una de las más empleadas debido a su efectividad para extraer las características más destacadas. En la siguiente Figura 3.6 se muestra el resultado de aplicar *max pooling* y *average pooling* sobre la misma matriz de entrada.



**Figura 3.6:** Operación pooling para una matriz 4x4 con filtro de pooling con tamaño 2x2.

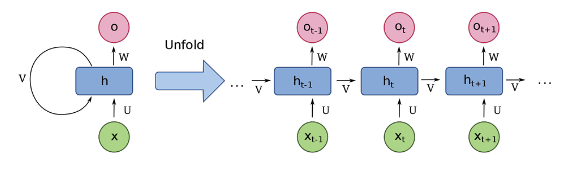
* La capa Fully Conected o Dense están ubicadas al final de la red, donde cada neurona en una capa dense, está conectada a todas las neuronas de la capa anterior, obteniendo como parámetro de entrada las características extraídas, realizando la clasificación y determinando la categoría o clase mediante una función de activación convirtiendo las salidas en probabilidades o a la clase que pertenece la entrada original.

Actualmente, este tipo de redes se emplean en aplicaciones cotidianas como Google Photos, Face ID, Instagram y Google Search.

### 3.5.2 Redes Neuronales Recurrentes

Las RNN está diseñadas principalmente para analizar secuencias de datos, donde la ordenación y dependencia tanto espacial como temporal son importantes. La información en las RNN fluye en bucle, es decir, la salida en un momento dado se procesa como entrada en el siguiente momento.

Una característica clave en las RNN es su capacidad de almacenar la información de los pasos anteriores mediante al denominado, estado oculto. Existen dos Redes Neuronales, de Corto y Largo plazo.



**Figura 3.7:** Arquitectura RNN (<https://kwfoundation.org/blog/2021/07/13/redes-neuronales-recurrentes/>)

Las Redes Neuronales de Memoria a Corto Plazo permite capturar dependencias en los datos, donde la influencia del estado oculto decae a medida que aumenta el tiempo, por lo que los pasos más próximos generarán un mayor impacto en la salida.

Las Redes Neuronales de Memoria a Largo Plazo aparecen para eliminar la limitación de capturar dependencias en los datos en las RNN estándar debido al desvanecimiento del gradiente, que se vuelven pequeños y dificulta el aprendizaje a medida que transcurre el tiempo. Aparecen dos soluciones: LSTM y GRU.

Para entender la Figuras 3.9 y la Figura 3.10, debemos de tener en cuenta la siguiente figura (Figura 3.7):

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Figura 3.8:** Leyenda para arquitecturas de la Figura 3.9 y Figura 3.10 (<https://www.themachinelearners.com/modelos-secuencia/>)

* **LSTM:** Diseñadas para manejar dependencias a largo plazo mediante celdas de memoria, que almacenan información durante un periodo prolongado. Esta celda está controlada por 3 puertas: Puerta de entrada, puerta de olvido y puerta de salida, que información agregan, eliminan o generan de la celda de memoria.

Diagrama, Esquemático

Descripción generada automáticamente

**Figura 3.9:** Arquitectura LSTM (<https://www.themachinelearners.com/modelos-secuencia/>)

* **GRU:** Diseñada para resolver el problema del desvanecimiento de gradiente en las RNN, considerándose una simplificación de LSTM. Emplean dos puertas, la puerta de actualización que decide cuanta información pasada debe de transmitir en el siguiente paso y la puerta de reinicio que se emplea para decidir cuanta información pasada se debe olvidar.

Diagrama, Esquemático

Descripción generada automáticamente

**Figura 3.10:** Arquitectura GRU (<https://www.themachinelearners.com/modelos-secuencia/>)

Esta capacidad de procesamiento de datos secuenciales hace que este tipo de red sea útil para aplicaciones como reconocimiento de voz, traducción de texto y generación de texto. Algunos ejemplos de aplicaciones son Google Translate, Asistentes virtuales y OpenAI GPT-2.

### 3.5.3 Redes Generativas Adversarias

Las GAN son un tipo de redes introducidas en 2014 y se emplean para que un cierto dato se adopte para otro conjunto de datos, es decir, busca que dada una entrada se asemeje demasiado a la distribución deseada y que el dato generado sea indistinguible comparado con un dato real de dicha distribución.

Las GAN están compuestas por dos ANN que compiten entre sí:

* Red generativa: Emplea el ruido aleatorio como entrada y genera datos falsos muy semejantes a los reales, es la que se encarga del trabajo creativo y se ve obligada a mejorar para conseguir engañar a la segunda ANN (Red discriminadora) o que le apruebe la tarea. Esta red es capaz de realizar millones de pruebas hasta que la Red discriminadora acepta el resultado.
* Red discriminadora: Tom datos reales como dato de entrada y trata de distinguir entre ellos, revisando las creaciones de la primera red y moderándola. Esta es mucho más precisa ya que a la IA actualmente se le da mejor retocar y reconocer que crear.

En la figura 3.10 se muestra la arquitectura de una GAN:

Diagrama

Descripción generada automáticamente

**Figura 3.10:** Arquitectura de una GAN (<https://www.xataka.com/inteligencia-artificial/conceptos-inteligencia-artificial-que-gans-redes-generativas-antagonicas>)

Al inicio del entrenamiento los datos generados por la Red generativa serán significantemente distintos a los datos reales, entonces no será complicado para la Red discriminadora, distinguir los datos reales de los generados y el error en ambos casos será diminuto.

Sin embargo, en la etapa final del entrenamiento, la Red generativa puede ser capaz de generar datos muy realistas y la Red discriminadoras se vuelve extremadamente eficiente, aunque ya no puede distinguir claramente entre datos reales y generados, debido a la alta capacidad creativa de la Red generativa. Este equilibrio entre la generación y discriminación hace que este tipo de redes sean muy útiles en aplicaciones para generar datos realistas, como imágenes, videos, texto, sonido y otros varios.

# 4. Estado del arte

En la actualidad el reconocimiento visual de aves utilizando DL ha avanzado notablemente gracias a los desarrollos de las CNN, aplicaciones y conjuntos de imágenes etiquetados. En este apartado se abordarán los métodos, tecnologías, arquitecturas y aplicaciones que se emplean en la actualidad para el reconocimiento visual de aves mediante imágenes.

## 4.1 Cornell Lab of Ornithology

El laboratorio de ornitología de Cornell se fundó en 1910 en Ithaca, Nueva York, Estados Unidos por unos miembros de la universidad que se centra en el estudio de aves y otros animales salvajes, asi como la naturaleza.

El principal objetivo de Birds Cornell es apoyar iniciativas que ayuden a la conservación de las aves y la educación en las ciencias y acciones por el medio ambiente para que impacten positivamente en la biodiversidad e inspirar a las personas, sin importar las edades y el sector que desempeñen, a cuidar la naturaleza y su entorno. Para ello, ofrece una gran cantidad variada de recursos públicos con los que se podrán obtener gran cantidad de datos, desde imágenes, audios, videos y seminarios, hasta aplicaciones para identificar aves mediante imágenes y sonido.

Unos ejemplos de aplicaciones sobre el reconocimiento de aves mediante imágenes son los siguientes:

* **eBird:** base de datos que contiene millones de datos sobre aves.
* **Merlin Bird ID:** herramienta diseñada para identificar aves.
* **Bird Academy:** Imparten seminarios creados para poder entender y conectar con las aves.
* **Living Bird:** es una revista trimestral que publica artículos sobre investigaciones y conservación de las aves.

En los siguientes apartados se profundizará en eBird y Merlin Bird ID.

### 4.1.1 eBird

ebird empezó como una idea simple en el laboratorio de Ornitología de Cornell hasta convertirse en uno de los proyectos científicos de biodiversidad más grande del mundo, gracias a su expansión en 2010.

Actualmente con el Gobierno de España, específicamente con el Centro Nacional de Educación Ambiental (CENEAM) donde el objetivo principal es la conservación y educación de las especies.

Imagen de la pantalla de un celular con la imagen de un ave

Descripción generada automáticamente con confianza baja

Figura 4.1: Inicio de la página web de eBird (<https://ebird.org/explore>)

eBird tiene una base de datos de observaciones sobre las aves, con un conteo de más de 100 millones de imágenes aportadas por científicos, ornitólogos y aficionados de todo el mundo. Estos datos no son solo imágenes, audios y videos, sino también la distribución, abundancia, habitat, comportamientos, localización y muchos más.

Esta base de datos es abierta y gratuita a aficionados, ebird cuenta con expertos en aves que realizan una rigurosa tarea de chequeo para garantizar la precisión, relevancia y la veracidad de los datos proporcionada por observadores.

En resumen, gracias a este proyecto se ha revolucionado la forma en la que se obtienen y emplean los datos sobre las aves y ha generado una comunidad global de observadores y científicos que contribuyen a la conservación de las especies y la biodiversidad.

### 4.1.2 Merlin Bird ID

Merlin Bird ID es una aplicación móvil gratuita implementada por el Laboratorio de Ornitología de Cornell que permite reconocer aves mediante imágenes, audios y descripciones de su aspecto, tamaño y comportamiento.



**Figura 4.2:** Logo de Merlin Bird ID (<https://earthsky.org/earth/media-we-love-merlin-bird-id-app-cornell-lab-review/>)

En las siguientes Figura 4.3, Figura 4.4, Figura 4.5 y Figura 4.6 mostramos el funcionamiento de la aplicación y los pasos que sigue:

* **Paso a paso:** reconoce aves mediante tamaño, color y comportamiento y genera un listado de las posibles aves que tienen esas características.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 4.2: Proceso para identificar un ave mediante la funcionalidad Paso a paso.

Un pájaro con la boca abierta

Descripción generada automáticamente con confianza media

Figura 4.3: Resultado de Paso a paso.

* **Audio:** reconoce aves mediante un audio en tiempo real y aparece una alerta cuando lo ha detectado.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Figura 4.4: Proceso para identificar un ave mediante audio.

* **Foto:** reconoce aves mediante imágenes en tu dispositivo o en tiempo real, puedes indicar la ubicación y la fecha o que no la sabes.

Imagen de la pantalla de un ave

Descripción generada automáticamente con confianza media

Figura 4.5: Proceso de reconocimiento de aves mediante fotos.

Actualmente, Merlin Bird ID es una de las mejores aplicaciones para el reconocimiento de aves, tanto visualmente como por audios. Es muy completa y precisa en sus análisis, gracias a su extensa base de datos y tecnología avanzada de IA.

## 4.2 iNaturalist

iNaturalist es una plataforma tanto web como móvil que fue desarrollada en el año 2008 como proyecto final de postgrado de Nate Agrin, Jessica Kline y Ken-ichi para la Academia de Ciencias de California en Berkeley. Permite a los usuarios llevar un registro de la biodiversidad y compartirlas a nivel mundial.

Los objetivos de esta plataforma es promover la ciencia ciudadana, recopilar datos para crear una base de datos global para proteger la biodiversidad y a su vez emplearse para conservacionistas, educadores e investigadores, aumentando el conocimiento público sobre la concienciación ambiental.

El reconocimiento automático de aves no es su única funcionalidad, ofrece una amplia cantidad de funcionalidades que no están limitadas a aves, pero es la que interesa en este TFG. En el Apartado (por definir) se profundizará en las funcionalidades específicas de reconocimiento visual de aves y a continuación, mencionaré algunas de las funcionalidades que tiene esta plataforma:

* **Registrar observaciones:** permite que los usuarios suban imágenes, audios y videos desde plantas y hongos hasta insectos o animales, con una serie de características como podría ser la ubicación precisa.
* **Comentarios:** los usuarios tienen la posibilidad de comentar y discutir sobre las observaciones registradas, proporcionando correcciones o ampliando la información sobre la especie observada.
* **Proyectos:** los usuarios tienen laposibilidad de crear y unirse a proyectos sobre cualquier especie en específico.

**Interfaz de usuario gráfica

Descripción generada automáticamente**

Figura 4.6: Funcionalidad de creación de un proyecto en la web de iNaturalist.

* **Herramienta de exportación y análisis:** los usuarios pueden exportar datos en formato CSV con gran cantidad de información pudiendo filtrar para limitar los datos extraídos y se puede visualizar un mapa de distribución geográfica de las observaciones registradas por los usuarios.

En resumen, iNaturalist es una plataforma gratuita muy poderosa para cualquier persona interesada en la conservación y el estudio de la biodiversidad, abarcando gran variedad de especies, no solo de aves.

# 5. Metodología

En este apartado se describirán las tecnologías empleadas para el desarrollo de este TFG, el software, hardare, tanto el dataset de prueba como el definitivo y las ANN empleados.

## 5.1 Hardware

Para la realización de este TFG, se ha empleado un ordenador de sobremesa y los recursos proporcionados por Google Colab.

Las especificaciones del dispositivo más importantes del ordenador de sobremesa son las siguientes:

* **CPU:** Intel(R) Core (TM) i5-10400 CPU @ 2.90GHz 2.90 GHz
* **RAM instalada:** 15.91 GB
* **Tipo de sistema:** Sistema operativo de 64 bits, procesador basado en x64
* **GPU:** NVIDIA GeForce RTX 2060
* **Versión del controlador:** 555.85

Respecto a las especificaciones de Windows son:

* **Edición:** Windows 10 Home
* **Versión:** 22H2
* **Compilación del sistema operativo:** 19045.4474

En la siguiente Tabla 5.1 se muestran las GPUs, CPUs y TPU que ofrece Google Colab Pro para la ejecución en entornos con Python:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre | Tipo Acelerador | RAM del sistema (GB) | RAM (GB) | GPU | Disco (GB) |
| CPU | CPU | Depende del modelo | Depende del modelo | No | Depende del modelo |
| TPU | TPU v4-32 | 32 | 32 | TPU v4-32 | 200 |
| A100 | A100 | 32 | 16 | NVIDIA Tesla A100 | 120 |
| V100 | V100 | 32 | 16 | NVIDIA Tesla V100 | 100 |
| T4 | T4 | 32 | 16 | NVIDIA Tesla T4 | 100 |

Tabla 5.1: GPUs CPUs y TPUs que ofrece Google Colab Pro.

## 5.2 Software

### 5.2.1 Python

Python es un lenguaje de programación de alto nivel en la que se le da mucha importancia a la legibilidad del código. Se trata de un lenguaje interpretado, es decir, que el código escrito no se traduce realmente a un formato legible en tiempo de ejecución.

Dibujo de una tortuga

Descripción generada automáticamente con confianza media

Figura 5.1: Python para IA.

Python es uno de los lenguajes más empleados para programar IA de reconocimiento de imágenes y el más empleado actualmente y con bastante diferencia, gracias a sus robustas bibliotecas y frameworks existentes, como TensorFlow, OpenCV, Keras, PyTorch, entre otros, la comunidad activa de desarrolladores que comparten sus conocimientos y soluciones y la gran flexibilidad que tiene para diferentes métodos de desarrollo y tipos de proyectos.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 5.2: Ranking de lenguajes más empleados en Mayo 2024 (<https://www.tiobe.com/tiobe-index/>)

Actualmente, empresas líderes como Google, Amazon y Netflix están empleando la combinación de Python e IA para impulsar la mejora e innovación de la eficiencia. Incluso una de las aplicaciones más revolucionarias de estos últimos años, ChatGPT, está mayormente programada en Python junto a otros lenguajes.

La versión empleada es 3.8.18.

### 5.2.2 Google Colab con Python

Google Colab es un servicio alojado de Jupyter Notebook que no requiere de ningún tipo de configuración y el acceso a recursos como CPU y GPU es completamente gratuito hasta un cierto límite de horas diario.

Google Colab ofrecen suscripciones de pago en la que obtienes unidades computacionales que se consumen dependiendo del tiempo que se ejecutan entornos. Existen diversos planes de pago dependiendo de lo que requiera cada usuario, así como obtener unidades informáticas sueltas.

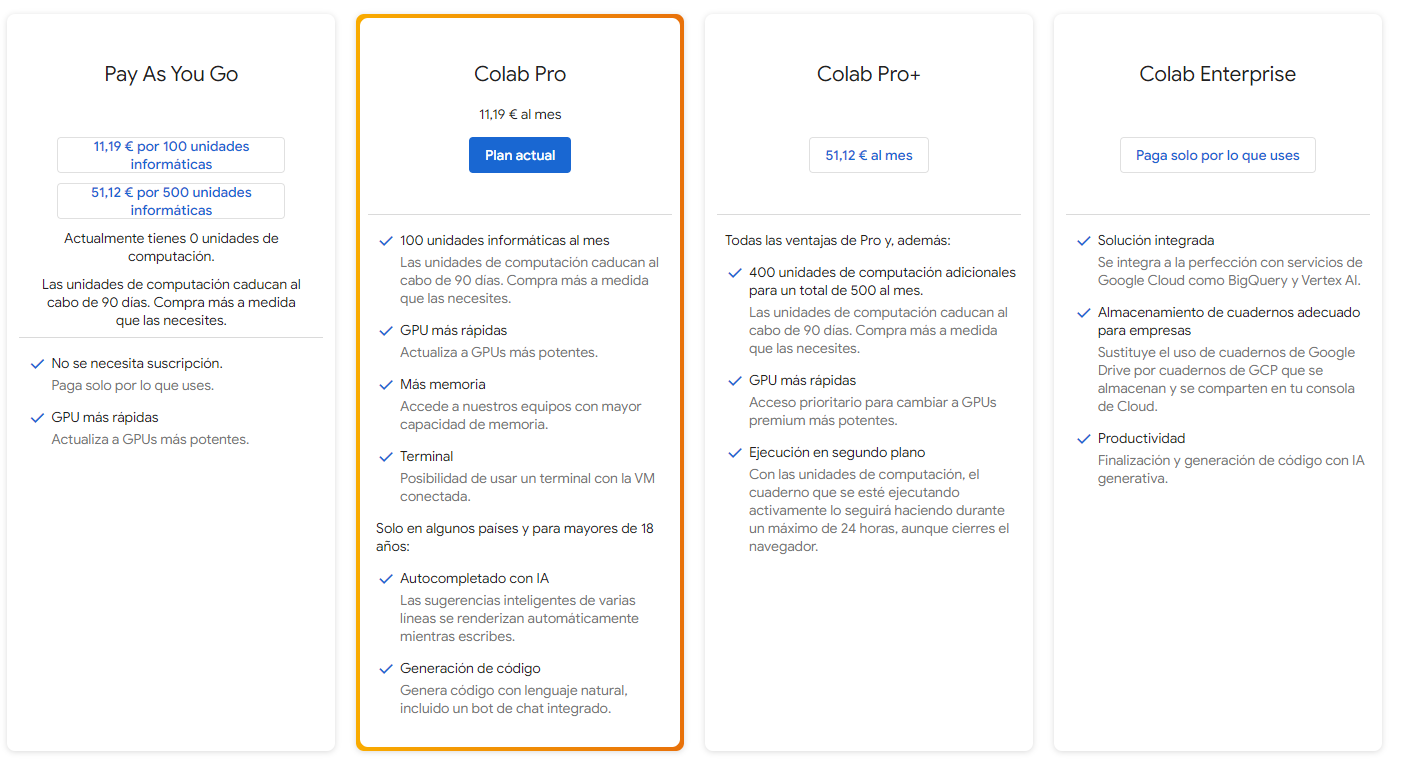


Figura 5.2: Planes de Google Colab.

La versión de Python instalada en Google Colab es 3.10.12

Google Colab es una herramienta que ha tomado mucha relevancia para investigadores y desarrolladores en el campo de la IA, específicamente en modelos de aprendizaje profundo, el procesamiento de imágenes y el análisis de datos.

### 5.2.3 Anaconda Distribution con Python

Anaconda fue creado especialmente para facilitar el uso de Python al análisis de datos empresariales. Desde entonces, el uso del lenguaje de Python ha incrementado con creces siendo el lenguaje mas popular hasta el momento como se muestra en la Figura 5.2 en el apartado anterior.

Anaconda es una distribución de Python basado en código abierto que funciona como un gestor de entorno, de paquetes y que 720 de ellos son de código abierto.

Logotipo, nombre de la empresa

Descripción generada automáticamente

Figura 5.3: Logo Anaconda.

Anaconda Distribution se distribuye en cuatro sectores que se instalan automáticamente y de manera sencilla, para obtener así una aplicación multiplataforma, Anaconda Navigator, Anaconda Project, Librerías de Ciencia de datos y Conda.

Las ventajas que tiene son múltiples. Permite instalar y administrar paquetes, dependencias y entornos de manera muy sencilla en con Python. Además, permite la creación de proyectos empleando múltiples entornos de desarrollo como JupyterLab, Jupyter Notebook, Spyder, VSCode, RStudio, entre otros, como se muestra en la Figura 5.4.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 5.4: Inicio de Anaconda Navigator.

Además de las ventajas comentadas anteriormente, la que interesa para este TFG es que permite compilar Python para una ejecución rápida, facilita la escritura de algoritmos complejos, cuenta con soporte para computación de alto rendimiento y elimina los problemas de dependencia entre los paquetes y el control de versiones. No obstante, siempre es recomendable asegurar las versiones correspondientes a sus dependencias revisando la documentación e instalarlas manualmente.

Las versiones instaladas son las siguientes:

* **Anaconda:** 2024.02-1
* **Python en el entorno:** 3.8

### 5.2.4 CUDA ToolKit y cuDNN

El kit de herramientas NVIDIA CUDA permite aprovechar la potencia de las GPU NVIDIA proporcionando un entorno de desarrollo para crear aplicaciones de alto rendimiento aceleradas por GPU que permite desarrollar, optimizar e implementar aplicaciones de aprendizaje automático, supercomputadoras, y otras más.

Las GPUs de NVIDIA están diseñadas específicamente para ejecutar tareas de manera simultánea, es decir, en paralelo que consiste en miles de núcleos de procesamiento pequeños pero lo suficientemente eficientes estando optimizados para manejar tareas en paralelo, lo que las hace ideales para sistemas que requieren de gran número de cálculos.

Respecto a las herramientas y librerías que ofrece NVIDIA CUDA Toolkit son muy numerosas y fundamentales para el procesamiento de alto rendimiento y baja latencia, específicamente diseñadas para el uso con CUDA. Una de las más importantes en el reconocimiento visual de imágenes es cuDNN (NVIDIA Deep Neuronal Network Library) que es una biblioteca primitiva para las DNN, proporcionando implementaciones altamente optimizadas para desarrollos de forward y backward convolucionales, pooling y normalización.

Un dibujo de una persona con una señal

Descripción generada automáticamente con confianza media

Figura 5.4: Logo NVIDIA CUDA y cuDNN.

Esta plataforma (NVIDIA CUDA) y librería (cuDNN) se pueden emplear junto a la aplicación Anaconda Navigator configurando un entorno de Python e instalando las librerías necesarias para aprovechar el aceleramiento por GPU. Esto es realmente efectivo en este TFG ya que el código emplea gran cantidad de cálculos, lo que te permite desarrollar y ejecutar código que se beneficie de la potencia de las GPU NVIDIA de una manera más integrada y sencilla.

Hay que tener en cuenta, mencionado anteriormente en el Apartado 5.1.3 Anaconda Distribution con Python, que las versiones del entorno creado en Anaconda Navigator con Python, de NVIDIA CUDA Toolkit y cuDNN tienen que ser compatibles. En las referencias [3] y [4] se muestran la compatibilidad de las versiones.

Las versiones instaladas son las siguientes:

* **NVIDIA CUDA Toolkit:** 11.2
* **cuDNN:** 8.1.0

### 5.2.5 Librerías

Todos los scripts empleados para este TFG han sido programados con Python. Por tanto, las siguientes librerías pertenecen a este lenguaje, alguna más empleada que otra, pero todas prescindibles.

**Concurrent**

Concurrent se emplea para realizar lanzamiento de tareas en paralelo, pudiendo mejorar el rendimiento de nuestro script o programa. El submódulo empleado es concurrent.futures el cual provee una interfaz de alto nivel para realizar invocaciones de manera asíncrona. Estas invocaciones se pueden realizar mediante hilos o procesos independientes, dependiendo de las funciones o necesidades que se requieran.

Esta librería se emplea para realizar descargas de imágenes de manera concurrente mediante el uso de hilos empleando la función *ThreadPoolExecutor* la cual unirá a todos los hilos creados antes de que el intérprete termine la ejecución del programa.

**CSV**

CSV (Coma Separated Value) es uno de los formatos más importantes para la importación y exportación de datos ya sea para hojas de cálculo o bases de datos. El módulo csv implementa clases permitiendo la lectura y escritura en formato CSV de manera sencilla y eficiente.

**Cv2**

Cv2 hace referencia a OpenCV, una librería de VC de código abierto ofreciendo una alta gama de funciones para el procesamiento de imágenes, de video y muchos más. Profundizando más en el reconocimiento de imágenes, incluyen operaciones para la manipulación de pixeles, modificando el tamaño de las imágenes, convirtiendo los colores y una gran integración con bibliotecas de aprendizaje profundo, como Tensorflow y PyTorch.

**Pillow**

Pillow es una biblioteca empleada para la manipulación de imágenes, es decir, agrega capacidades de procesamiento de imágenes proporcionando una amplia compatibilidad de formatos de archivo y capacidades de procesamiento de imágenes bastante potentes. Es una actualización de la librería PIL.

Principalmente se emplea para abrir y cargar imágenes, aplicar filtros, detección de bordes, corrección de colores, procesar imágenes e incluso generar imágenes simples, como gráficos o diagramas y muchas funciones más.

**Keras**

Keras es una biblioteca de redes neuronales de alto nivel de código abierto escrita en Python que se ejecuta sobre frameworks como Theano y TensorFlow.

Está diseñada principalmente para que sea modular, rápida y sencilla de emplear, siendo una de las más empleadas en la actualidad ya que facilita enormemente la creación de capas en ANN o la configuración de distintas arquitecturas y cuenta con una gran comunidad de desarrolladores y usuarios que contribuyen en mejoras y documentación.

Una de sus mejores características es que está diseñada para aprovechar el poder de las unidades de GPU, pudiendo acelerar los procesos de modelos de ANN con gran cantidad de datos.

**Math**

Math es una biblioteca de las más sencillas y útiles ya que permite el acceso a funciones matemáticas y constantes que son empleadas en operaciones. Estas funciones van desde lo más básico como una suma o resta hasta funciones más complejas como redondeos, logaritmos, trigonometría, exponentes y muchas más.

**Matplotlib**

Matplotlib es una de las librerías más exhaustivas y populares para la visualización de datos en Python ya sean estáticos, animados o interactivos. Puede llegar a ser muy potente si se combinan con otras librerías como NumPy, Pandas, SciPy, Seaborn y otras más.

Permite la generación de una gran variedad de gráficos, desde una simple línea o de diagramas hasta modelos volumétricos y 3D como se muestra algunos tipos en [5]. También tiene una gran flexibilidad permitiendo a los usuarios personalizar y combinar diferentes tipos de gráficos, así como añadiendo etiquetas, leyendas, títulos, cambiando colores, estilos y formatos e incluso con el uso de bibliotecas como Ploty o Bokeh generar visualizaciones interactivas.

En resumen, es una de las bibliotecas más empleadas gracias a que tiene una gran flexibilidad y variedad de gráficos y visualizaciones, permitiendo mostrar los datos de manera sencilla y atractiva.

**Mlxtend**

Mlxtend (Machine learning extensions) es una librería muy útil para las tareas de data science o ML. Contiene gran variedad de funciones que permiten la evaluación de modelos, preprocesamiento de los datos y visualización de datos que complementan algunas funcionalidades de otras librerías como scikit-learn y matplotlib.

Se ha empleado para la generación de matrices de confusión para evaluar el rendimiento de los modelos de clasificación, que muestra una tabla con las comparaciones de las etiquetas originales con las etiquetas predicha, siendo de importancia para entender el rendimiento y la precisión del modelo entrenado.

**Numpy**

Numpy, abreviatura de *Numerical Python, es* una biblioteca creada en 2005 por Travis Oliphat para extender la biblioteca *Numeric* y que da soporte para la creación vectores y matrices multidimensionales junto a una gran colección de funciones matemáticas de alto nivel para poder operar con estas.

Incorpora una nueva clase de objetos llamados **arrays** que permite realizar colecciones de datos del mismo tipo con varias dimensiones y funciones muy eficientes para su manipulación. En la siguiente Figura 5.5 se muestra un ejemplo de las estructuras de los **arrays**, siendo axis las dimensiones del array o también conocidos como ejes.

Diagrama

Descripción generada automáticamente con confianza baja

Figura 5.5: Estructura de los arrays con distintas dimensiones.

La ventaja de esta biblioteca es el procesamiento de los arrays que la hacen una buena elección para el procesamiento de vectores y matrices ya que realiza las operaciones hasta cincuenta veces más rápido que las listas.

Actualmente, se ha vuelto una de las bibliotecas fundamentales en la IA, por su capacidad de procesamiento de matrices, en la ciencia de datos, análisis o cualquier área que requiera de cálculos complejos en Python y sus usos van desde extraer datos de un simple array, como el valor máximo o mínimo de un array, hasta operaciones complejas de algebra lineal, estadística y otras más.

**Os**

Os es una de las librerías más empleadas en Python. Permite interactuar con el Sistema Operativo, permitiendo funcionalidades dependientes de este, desde manejar directorios y contar la cantidad de elementos en la carpeta hasta

**Scikit**-**learn**

Sklearn es una biblioteca de software libre muy popular de Python que se emplea para el ML, siendo accesible para todos. Está construido sobre las bibliotecas NumPy, SciPy y matplotlib y ofrece múltiples herramientas simples y eficientes para el análisis predictivo de datos.

En la siguiente Figura 5.6 se muestra la función que crea una matriz de confusión que evalúa el rendimiento de un modelo de clasificación, mostrando la cantidad de predicciones correctas e incorrectas desglosadas por clases.

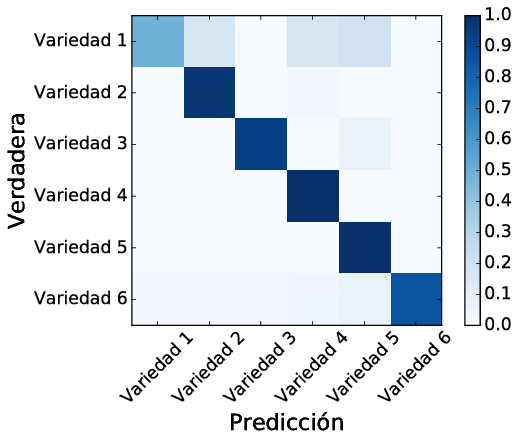


Figura 5.6: Ejemplo de la salida de una matriz de confusión.

Actualmente, es una biblioteca fundamental para la IA y ciencia de datos debido a la facilidad de uso y la amplia gama de algoritmos, desde algoritmos de regresión lineal y algoritmos de clasificación como K-nearest hasta algoritmos de reducción de dimensionalidad.

**TensorFlow**

TensorFlow es una biblioteca de código abierto creada en 2015 por Google Brain para el ML y DL. Empleando una arquitectura permite crear y entrenar redes ANN para detectar patrones que detectan los humanos.

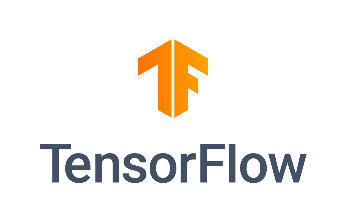


Imagen 5.7: Logo TensorFlow.

Es la herramienta que facilita la gestión e implementación de los procesos de ML y DL de forma que desarrollar una ANN sea de forma fácil, sin importar el lenguaje o la plataforma que utilices. Además de trabajar con ANN, TensorFlow es multiplataforma lo que hace posible trabajar con GPUs y CPUs e incluso con las unidades de procesamiento de tensores (TPUs).

Las utilidades de TensorFlow son extensas gracias a su robustez y capacidad de integridad, teniendo un repositorio para modelos preentrenados, llamado TensorFlow Hub, donde los desarrolladores o investigadores pueden reutilizar y adaptar dependiendo de las necesidades específicas que se requieran. Tiene una gran capacidad de manejar grandes volúmenes de datos y cálculos, siendo una de las métricas más importantes en el desarrollo de la IA la cual continuará creciendo debido a la gran demanda de soluciones para diversos sectores.

Actualmente, Tensorflow se emplea en una amplia gama de aplicaciones, desde el reconocimiento de imágenes y NPL hasta el análisis de sentimientos y diagnósticos médicos.

## 5.3 Dataset

El dataset seleccionado para las pruebas iniciales sobre los modelos es NABirds (North América Birds) que contiene una colección de imágenes de 48000 imágenes etiquetadas de 400 especies distintas que son muy comunes en Norte América, incluyendo entre ellas la diferenciación entre machos, hembras y crías, componiendo así la cantidad aproximada de 700 categorías distintas.

Las imágenes seleccionadas para el dataset se obtuvieron mediante la colaboración entre el Laboratorio de Ornitología de Cornell, Cornell Tech, Caltech, la Universidad de Brigham Young y miles de contribuyentes. Las imágenes seleccionadas fueron verificadas para que fuesen alta calidad y donde las aves fuesen claramente visibles e identificables, eliminando imágenes duplicadas y aquellas que no cumplían los requisitos de calidad necesarios. Una vez filtradas se realizaron anotaciones detallados, incluyendo la especie del ave, la ubicación de la foto y más aspectos relevantes para el reconocimiento visual de imágenes.

Todas las clases seleccionadas tienen la misma cantidad de imágenes, 150 imágenes por clase y tras filtrarlas y realizar las anotaciones se procesaron para estandarizar el formato, resolución y el tamaño.

En resumen, el dataset de NABirds se empleará durante las pruebas iniciales para evaluar el rendimiento de diversos algoritmos de reconocimiento visual, con el objetivo de mejorar la precisión y eficiencia para detectar aves.

Respecto al dataset para la detección de imágenes en la provincia de Alicante, que es lo que concierne en este TFG, se realizará un dataset propio con más de 12000 imágenes y 24 clases etiquetadas y cada una de ellas con un número variable de imágenes, mediante herramientas y aplicaciones gratuitas o con licencias gracias a la Universidad de Alicante. Sobre este dataset se profundizará en el próximo apartado 6.1 Dataset propio.

# 6. Desarrollo

En este apartado se describirá el proceso para llegar a la fase de resultados mostrados en el apartado 7. Para ello, se describirán los pasos seguidos en cada uno de los apartados enfocados en el reconocimiento visual de aves. Algunos apartados son de la obtención de los datos, las arquitecturas empleadas o la experimentación.

## 6.1 Obtención de datos para el dataset

En esta fase se profundizará sobre la obtención y análisis de los datos para la creación de un dataset propio. Se explicará el origen de los datos, diferentes formas de recopilación y las aplicaciones o herramientas empleadas, asi como *scripts* de ayuda.

6.1.1 Dataset

Para la selección de las especies de aves se hizo un análisis a los listados de 2024 de iNaturalist, eBirds y artículos sobre humedales alicantinos. Finalmente, tras analizar los 3 listados, se obtuvieron veinticuatro especies de aves para realizar el TFG.

En la siguiente Figura 6.10 se muestra las clases de aves escogidas para la creación de un dataset propio.

Tabla

Descripción generada automáticamente

Figura 6.1: Nombre de las especies de las aves.

En el siguiente apartado 6.2 Análisis de datos, se profundizará en el estudio del dataset, su división de los conjuntos, así como las ventajas y desventajas que asociadas.

### 6.1.2 Recopilación de datos

La recopilación de datos es una parte fundamental para el desarrollo de cualquier proyecto de análisis y modelado de datos, ya que la calidad de los datos influye directamente en la precisión y efectividad del modelo a desarrollar. En el contexto de este TFG enfocado al reconocimiento visual de imágenes hay que prestar atención en aspectos clave como el etiquetado, calidad y la veracidad de la fuente de los datos.

**Origen de los datos**

Existen diversas herramientas de acceso público enfocadas a la recopilación de imágenes de aves, entre otras, que ofrecen colecciones extensas de imágenes etiquetadas por expertos. La mayoría de las imágenes obtenidas provienen de la plataforma de iNaturalist y una minoría de eBirds.

iNaturalist contiene una herramienta para exportar observaciones de aves, entre muchas otras, que solo son accesibles si te registras como usuario. Esta herramienta contiene un apartado de consultas para filtrar las observaciones ajustándose a la necesidad del usuario. Entre ellos se encuentran los más importantes para este proyecto, que solo incluya fotos, que estén etiquetadas como ave, el taxón o nombre de la especie y a ser posible que esté revisada, y si queremos profundizar en un lugar específico se puede indicar el continente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 6.2: Apartado de consulta de observaciones iNaturalist.

Una vez realizada la búsqueda, nos mostrará un listado de vista previa del ave consultada y otro apartado más para exportar los datos a un CSV.

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Figura 6.3: Apartado selección de columnas para exportar a CSV.

Como se muestra en la Figura 6.3, para obtener las imágenes solo necesitamos la url de la imagen. Existen muchas más columnas aparte de las mostradas, pero para este proyecto son irrelevantes.

Una vez obtenemos el CSV, hay que descargar las imágenes de la columna exportada *image\_url*, para ello se procede a la creación de un *script* en Python. Se podría realizar manualmente, pero teniendo en cuenta que se exportan más de 900 imágenes en algunas clases de aves, es mejor descartar esta idea y programar un *script* que nos ayude con la descarga de estas imágenes.

Este *script* se ha realizado en Python empleando librerías de paralelismo, para que el tiempo del proceso de disminuyese. La librería empleada en este caso ha sido concurrent.futures nombrada en el apartado *5.2.5 Librerías*. Este script creaba la carpeta con el nombre de la especie del ave en caso de no existir y almacenaba las imágenes descargadas del fichero CSV.

Respecto a la extracción de imágenes mediante eBird, se hizo manualmente para completar algunas clases añadiendo características o puntos de vista diferentes que no nos proporcionaba iNaturalist por falta de datos en algunas especies, como por ejemplo crías, fotografías al vuelo y perspectivas de frente.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Figura 6.4: Ejemplo de consulta con filtros de eBird.

Para descargar las imágenes, se ha realizado la opción de Google de “*Guardar como*” en la carpeta correspondiente y renombrando la imagen, ya que eran una proporción muy pequeña de imágenes.

### 6.1.3 Filtrado de datos

Una vez descargadas todas las imágenes de iNaturalist, es necesario realizar una revisión de las imágenes para ver que realmente la especie descargada coincide con la esperada y que cumpla una serie de requisitos para que el modelo no tenga problemas en la fase de entrenamiento.

A continuación, se detallarán los filtros que se han aplicado para obtener el dataset definitivo.

**Clasificación taxonómica**

La clasificación taxonómica es una tarea compleja y propensa a errores, algunas imágenes están mal clasificadas debido a que no están revisadas y etiquetadas por un experto o no tienen la calidad suficiente para clasificarla. Este tipo de errores suele ocurrir cuando ciertas aves tienen características muy similares.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 6.5: Ejemplo de mala clasificación de un Serín Verdecillo.

**Cantidad de aves**

La cantidad de aves que aparecen en una imagen es un aspecto muy importante en el entrenamiento del modelo, puede ser beneficioso si contamos con una gran cantidad de datos sobre esa especie, ya que al entrenar aumenta la variabilidad de los datos de entrenamiento y permite identificar mejor a la especie, ya que algunas especies es muy complicado ver a una sola ave en la imagen.

Por el contrario, si se trata de un modelo sencillo como este TFG, incluir múltiples aves o multitudes, puede llevar al modelo a aprender características no relacionadas con el ave, datos redundantes o incluso llevar a un sobreajuste en lugar de mejorar y que dificultaría la clasificación del ave.



Figura 6.6: Ejemplo de multitud de flamencos común en los arrozales de l’Albufera.

Finalmente, para asegurar la calidad y precisión del modelo se toma la decisión de no incluir aquellas imágenes que incluyan a más de un ave en la imagen, ya sean múltiples aves o multitudes.

**Calidad de imagen**

La calidad de las imágenes es fundamental en la detección de imágenes, ya que cuanta mejor calidad tenga las imágenes mayor precisión tendrá el modelo, ya que podrán aprender mejores detalles y características de estas aves, sobre todo en aquellas aves que tengan mucha similitud.

En cambio, si no descartamos imágenes borrosas, el ave muy al fondo de la imagen o de baja resolución pueden hacer que el modelo tenga dificultades para identificar patrones relevantes que identifican claramente al ave.



Figura 6.7: Ejemplo de imagen borrosa de un Jilguero europeo.

En resumen, las imágenes de mala calidad se descargan del modelo ya que es un aspecto que afecta a todas las etapas del entrenamiento del modelo.

**Sexo y etapa de desarrollo**

En la mayoría de las especies obtenidas en la recopilación de datos, contienen imágenes de diferentes sexos y etapas de desarrollo. Esto es muy importante ya que la mayoría de las aves cambian considerable mente entre las etapas de cría y adultez, y en algunas de ellas hay diferencias muy notables entre los sexos.

Por ello, se decide descartar aquellas imágenes de cría, machos o hembras que no tengan una gran proporción del conjunto de datos de esa clase de ave ya que habría un desbalance y no identificaría bien las características de esas imágenes.



Figura 6.8: Diferencia entre macho y hembra de Pato cucharon norteño (arriba macho y abajo hembra).

Aunque se han descartado, existe una solución para no descartarlas, sería separar los sexos o crías en otra clase y realizar aumento de datos creando diferentes versiones modificadas de las imágenes existentes.

**Plumaje y huevos**

Las imágenes de plumas y huevos generan una situación muy similar a la anterior con crías y aves adultas ya que se requiere que un gran porcentaje del conjunto de esa clase sean plumas y/o huevos.

Las imágenes exportadas no tienen gran cantidad de imágenes de plumas y algunas de las clases son nulas en ese aspecto. Respecto a las imágenes de huevos, solo hay algunas clases que tienen una cantidad de imágenes considerable.

Imagen que contiene pasto, heno, nido, pequeño

Descripción generada automáticamente

Figura 6.9: Huevos de chorlito dorado europeo.

Por ello, se procede a descartar las imágenes de plumaje e incluir solo en algunas de ellas las imágenes de huevos.

**Comportamiento**

Una de las características más importantes dentro del reconocimiento de aves, es el comportamiento especifico de ellas. La mayoría de las aves tienen comportamientos muy diferentes entre ellas, lo que las hace más distinguibles de sí.

Estos comportamientos no solo se basan en acciones, sino que también puede ser en el entorno en el que se rodean, forma de alimentarse, canto, etc. Este tipo de comportamientos pueden ser captados en cámara y empleados como herramientas efectivas para la clasificación del tipo de ave.



Figura 6.10: Pato cucharon norteño extendiendo las alas para secarse y regular su temperatura.

En nuestro conjunto de imágenes se han incluido imágenes de comportamientos de las aves, ya sean aleteos, forma de las alas en el vuelo, de alimentación o la manera de beber agua, postura en reposo o incluso el cortejo antes del apareamiento.

**Aves enfermas o muertas**

El motivo de incluir imágenes de aves enfermas o muertas es entrenar al modelo para que sea capaz de distinguir no solo aves sanas, sino también aquellas que presentan enfermedades que comprometen su integridad física, cambiando la forma visual, o de aquellas que han fallecido, pero que aún pueden ser clasificadas sin problema. No todas las clases incluyen este tipo de imágenes, ya que en algunas de ellas no existen o no hay suficientes imágenes para que suponga una mejora en el modelo.

Además, hay que destacar que no se han incluido imágenes de huesos o aves aplastadas.

## 6.2 Análisis de datos

Una vez terminada la extracción y el filtrado de las imágenes, en este apartado se analizará el dataset propio y se abordarán aspectos como la cantidad de imágenes, la distribución de las clases y la división en conjuntos de entrenamiento, validación y de prueba.

### 6.2.1 Cantidad de imágenes y distribución

En este apartado se mostrará la distribución de las clases y la cantidad de imágenes para cada una de las clases.

En la Figura 6.11 se muestra un diagrama de barras con la cantidad de imágenes, eje vertical, por cada clase ordenadas alfabéticamente, eje horizontal.

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figura 6.11: Diagrama de barras de la cantidad de imágenes por clase.

Como se puede observar en la figura anterior, todas las tienen una gran cantidad de imágenes, predominando la gaviota patiamarilla, pero claramente hay tres clases que destacan por la baja cantidad de imágenes. Estas son la gaviota picofina con ochenta y cuatro imágenes, seguida del vencejo común con ciento sesenta y pardela balear con ciento sesenta y una imagen.

Este problema se conoce como desbalanceamiento de imágenes, pudiendo afectar negativamente al rendimiento del modelo, pudiendo disminuir la precisión en estas clases debido a la falta de ejemplos de entrenamiento. Además, el modelo puede sufrir sobreajustes por la gran diferencia de cantidad de imágenes, generando una falta de generalización, no pudiendo adquirir el correcto aprendizaje de las características de esa clase.

Respecto a las clases restantes, hay que destacar que tienen una cantidad considerable de imágenes lo que hace que el aprendizaje del modelo sea más robusto y generalizable. Con estas cantidades de imágenes el modelo tiene menos probabilidad de sufrir un sobreajuste o más conocido como overfitting, aprender y discriminar características de cada clase y mejorar la precisión de la clasificación.

En resumen, el dataset o conjunto de imágenes puede considerarse bueno en términos generales debido a la cantidad total de imágenes y a la cantidad de clases. Hay que tener en cuenta las limitaciones de las clases con baja cantidad de imágenes y elaborar estrategias para solventar este tipo de problemas durante el entrenamiento del modelo.

6.2.2 Inconvenientes del dataset para el modelo

### 6.2.3 División del conjunto

## 6.3 Modelos

## 6.4 Experimentación

Resultado EfficientB7: No he podido ejecutar con esta arquitectura porque necesitaría 32 GB aproximadamente.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

# 7. Resultados

# 8. Conclusión

# Bibliografía

Inteligencia artificial:

<https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr>

<https://datascientest.com/es/inteligencia-artificial-definicion#:~:text=La%20historia%20de%20la%20inteligencia%20artificial%20comenz%C3%B3%20en%201943%20con,creaci%C3%B3n%20de%20una%20red%20neuronal>

<https://www.youtube.com/watch?v=_pcQnFz7wAI&ab_channel=RaicesdeEuropa>

<https://www.nationalgeographic.com.es/ciencia/breve-historia-visual-inteligencia-artificial_14419>

<https://medium.com/@natisr/historia-de-la-inteligencia-artificial-63277f78fe2c>

<https://cse.unl.edu/~choueiry/S09-476-876/Documents/whatisai.pdf>

<https://www.algotive.ai/es-mx/blog/historia-de-la-inteligencia-artificial-el-machine-learning-y-el-deep-learning#ancla_3>

<https://www.bbva.com/es/innovacion/machine-learning-que-es-y-como-funciona/>

IA, DEEP LEARNING Y MACHINE LEARNIG:

<https://intersog.co.il/blog/ais-powerful-duo-difference-between-machine-learning-and-deep-learning/>

Tipos algoritmos DEEP learning:

<https://www.tokioschool.com/noticias/tipos-deep-learning/>

Desventajas del Deep learning:

<https://prompt.uno/redes-neuronales-profundas/desventajas-de-las-redes-neuronales-profundas/>

Redes neuronales artificiales:

<https://www.unir.net/ingenieria/revista/redes-neuronales-artificiales/>

Transfer learning:

<https://www.tokioschool.com/noticias/transfer-learning/>

<https://www.mdpi.com/1424-8220/23/2/570>

<https://www.geeksforgeeks.org/transfer-learning-with-fine-tuning-in-nlp/>

CNN: <https://datascientest.com/es/convolutional-neural-network-es>

<https://medium.com/latinxinai/redes-neuronales-convolucionales-cnn-dd4b50ceb24>

<https://paperswithcode.com/method/max-pooling>

RNN: <https://torres.ai/redes-neuronales-recurrentes/>

<https://datascientest.com/es/recurrent-neural-network-rnn-de-que-se-trata>

<https://keepcoding.io/blog/que-son-las-redes-neuronales-recurrentes/>

<https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/>

https://kwfoundation.org/blog/2021/07/13/redes-neuronales-recurrentes/

<https://gamco.es/glosario/red-neuronal-recurrente/>

GAN:

<https://arxiv.org/pdf/2302.09346>

<https://blog.damavis.com/redes-neuronales-generativas-adversarias-gans/>

<https://arxiv.org/abs/1406.2661>

<https://www.deeplearningbook.org/>

[https://www.allaboutbirds.org/news/living-bird-latest-issue/#](https://www.allaboutbirds.org/news/living-bird-latest-issue/)

Estado del arte:

<https://keras.io/api/applications/efficientnet/>

<https://merlin.allaboutbirds.org/es/pagina-de-inicio/>

Cornell:

<https://www.birds.cornell.edu/latam/programas-y-proyectos/>

<https://www.solucionescosteras.org/recursos-del-cornell-lab-of-ornithology/>

<https://www.miteco.gob.es/es/ceneam/quienes-somos.html>

<https://www.miteco.gob.es/es/ceneam/recursos/pag-web/ebird.html>

<https://www.miteco.gob.es/es/ceneam/recursos.html>

NumPy

<https://numpy.org/>

<https://aprendeconalf.es/docencia/python/manual/numpy/>

Sklearn

<https://scikit-learn.org/stable/>

Ebird:

<https://media.ebird.org/catalog?taxonCode=eurgol&mediaType=photo>

Dataset Birds:

<https://www.inaturalist.org/>

<https://ebird.org/spain/home>

<https://www.vision.caltech.edu/datasets/cub_200_2011/>

Colab:

<https://colab.research.google.com/signup>

Python:

<https://eiposgrados.com/blog-python/por-que-python-mejor-lenguaje-para-programar-ia/>

<https://www.linkedin.com/pulse/python-y-la-inteligencia-artificial-combinaci%C3%B3n-perfecta-laguna/>

Anaconda

<https://heardlibrary.github.io/digital-scholarship/script/anaconda/>

CUDA:

https://developer.nvidia.com/cudnn

Aves:

<https://vadeaves434152954.wordpress.com/2024/02/14/humedales-alicantinos/>

# Referencias

[1] John McCarthy 1962. *LISP 1.5 programmer's manual.*

[2] [Online] INER (29/03/2022). Recuperado de <https://www.unir.net/ingenieria/revista/sistema-experto/#:~:text=Los%20sistemas%20expertos%20(SE)%20son,un%20profesional%20en%20la%20materia>

[3] NVIDIA. (n.d.). CUDA compatibility. NVIDIA Developer. Recuperado de <https://docs.nvidia.com/deploy/cuda-compatibility/index.html>

[4] Anaconda, Inc. (s. f.). Paquetes GPU. Documentación de Anaconda. Recuperado de <https://docs.anaconda.com/free/working-with-conda/packages/gpu-packages/>

[5] Matplotlib. (s.f.). Plot types - 3D and volumetric data. Recuperado de <https://matplotlib.org/stable/plot_types/index.html#d-and-volumetric-data>

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Codificando Bits. (2023, 13 de febrero). Autoencoders: explicación y tutorial en Python. Codificando Bits. Recuperado de <https://www.codificandobits.com/blog/autoencoders-explicacion-y-tutorial-python/>

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[Online] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946v5. Recuperado de https://arxiv.org/abs/1905.11946v5

# Lista de Acrónimos y Abreviaturas

ANN Redes Neuronales Artificiales

CNN Redes Neuronales Convolucionales

CPU Central Processing Unit

CSV Valores Separados Por Comas

CUDA Compute Unified Device Architecture

cuDNN CUDA Deep Neural Network

DL Deep Learning

DNN Redes Neuronales Profundas

GAN Redes Generativas Adversarias

GPU Graphic Processing Unit

GRU Unidades Recurrentes Cerradas

IA Inteligencia Artificial

LSTM Memoria a largo plazo

ML Machine Learning

NLP Procesamiento de Lenguaje Natural

ReLU Unidad Lineal Rectificada

RNN Redes Neuronales Recurrentes

TFG Trabajo de Fin de Grado

TPU Tensor Processing Unit

VC Visión por computadora