



Escuela
Politécnica
Superior

Reconocimiento visual de aves con Deep Learning



Grado en Ingeniería en Sonido e Imagen
en Telecomunicación

Trabajo Fin de Grado

Autor:

Alberto Nicolás Montesinos Torregrosa

Tutor/es:

José García Rodríguez

Esther Sebastián González

Julio 2022



Universitat d'Alacant
Universidad de Alicante

Resumen

En los últimos años la inteligencia artificial no ha parado de crecer, consiguiendo incluso crear máquinas que aprendan como los humanos.

Por ello uno de los retos del aprendizaje automático es aproximarse a la percepción y razonamiento humano, para así poder clasificar las imágenes por los objetos que presentan. En el mundo de las aves, un clasificador permitiría mecanizar un trabajo muy costoso y que requiere de muchos recursos, lo cual supone un gran problema.

En este proyecto se ha trabajado en la implementación de un clasificador de aves mediante redes neuronales convolucionales. Primero se han adquirido los conceptos básicos y necesarios para realizar este trabajo, para posteriormente crear una base de datos propia y un modelo capaz de distinguir entre diferentes especies de aves.

Por último, una vez concluida la implementación se presentan las conclusiones del trabajo y que posibles líneas se podrían afrontar en un futuro.

Palabras clave

Machine Learning, Deep Learning, Transfer Learning, Redes Neuronales Convolucionales, dataset, clasificación de imágenes.

1 Contenido

| | | |
|-------|---|----|
| 1. | Introducción..... | 11 |
| 1.1 | Motivación..... | 11 |
| 1.2 | Propuesta y objetivos..... | 12 |
| 1.3 | Estado del arte | 13 |
| 1.3.1 | TheCornellLab | 13 |
| 1.3.2 | iNaturalist..... | 14 |
| 1.3.3 | Arquitecturas de aprendizaje profundo para reconocimiento visual. | 14 |
| 1.3.4 | Fuentes de Datos | 15 |
| 2 | Marco teórico | 17 |
| 2.1 | Inteligencia artificial..... | 17 |
| 2.2 | Machine Learning..... | 18 |
| 2.3 | Deep Learning | 20 |
| 2.4 | Redes neuronales artificiales | 20 |
| 2.5 | Red Neuronal Convolucional (CNN) | 21 |
| 2.5.1 | Capa convolucional | 22 |
| 2.5.2 | Pooling | 23 |
| 2.5.3 | Capas de activación | 24 |
| 2.5.4 | Capa Fully Connected: | 26 |
| 2.6 | Transfer Learning | 26 |
| 2.7 | Arquitecturas | 27 |
| 2.7.1 | VGGNet | 27 |
| 2.7.2 | Resnet50..... | 28 |
| 2.7.3 | Inception ResNet-v2..... | 29 |
| 2.7.4 | MobileNet..... | 29 |
| 2.7.5 | EficientNet | 30 |
| 2.8 | Overfitting | 31 |

| | | |
|-------|---|----|
| 2.9 | Tecnología | 32 |
| 2.9.1 | Software | 32 |
| 2.9.2 | Hardware | 34 |
| 3 | Implementación..... | 35 |
| 3.1 | Planteamiento inicial | 35 |
| 3.2 | Obtención de los datos..... | 35 |
| 3.2.1 | Recopilación de datos..... | 36 |
| 3.2.2 | Limpieza de datos..... | 36 |
| 3.2.3 | Especies de estudio..... | 39 |
| 3.3 | Análisis de los datos | 40 |
| 3.3.1 | Clases desbalanceadas..... | 40 |
| 3.3.2 | Similitud entre imágenes..... | 41 |
| 3.3.3 | Diferencia de calidad o distancia | 41 |
| 3.4 | Modelo..... | 43 |
| 3.4.1 | Arquitecturas usadas | 43 |
| 3.4.2 | Aumentado de datos | 44 |
| 4 | Experimentación..... | 45 |
| 4.1 | Nabirds | 45 |
| 4.2 | Base de datos propia | 48 |
| 4.3 | Pruebas | 50 |
| 5 | Conclusión..... | 53 |
| 6 | Referencias | 54 |

Índice ilustraciones

| | |
|--|----|
| Ilustración 1: Logo de Merlin Bird ID | 14 |
| Ilustración 2: iNatrulist | 14 |
| Ilustración 3: Camera trap | 16 |
| Ilustración 4: Diagrama de Venn de la Inteligencia artificial | 17 |
| Ilustración 5: Esquema ejemplo Machine Learning | 18 |
| Ilustración 6: Ejemplo Clasificación y Regresión | 18 |
| Ilustración 7: Ejemplo de Clustering | 19 |
| Ilustración 8: Esquema ejemplo aprendizaje..... | 19 |
| Ilustración 9: Esquema ejemplo Machine Learning | 20 |
| Ilustración 10: Esquema red neuronal artificial | 21 |
| Ilustración 11: Esquema CNN | 22 |
| Ilustración 12: Proceso capa convolucional | 22 |
| Ilustración 13: Ejemplo de Padding | 23 |
| Ilustración 14: procesamiento del Max Pooling | 24 |
| Ilustración 15: Función de activación Sigmoide..... | 24 |
| Ilustración 16: Función de activación Tangente hiperbólico | 25 |
| Ilustración 17: Función de activación ReLu(Redes Neuronales | 25 |
| Ilustración 18: Función de activación Softmax..... | 26 |
| Ilustración 19: Esquema ejemplo de Transfer Learning | 27 |
| Ilustración 20: arquitectura vgg16 | 28 |
| Ilustración 21: Arquitectura Resnet50 | 28 |
| Ilustración 22: Arquitectura inception-ResNet-v2 | 29 |
| Ilustración 23: Arquitectura MobileNet | 30 |
| Ilustración 24: Arquitectura EfficientNet..... | 31 |
| Ilustración 25: Método de extracción de datos | 36 |
| Ilustración 26: ejemplos varias aves en una foto | 37 |
| Ilustración 27: ejemplo de multitud de aves | 37 |
| Ilustración 28: Ejemplo pluma y huevo | 38 |
| Ilustración 29: Gráfico especies desbalanceadas | 40 |
| Ilustración 30: ejemplo similitud (Gaviota de Aduoin y Gaviota Patiamarilla) . | 41 |
| Ilustración 31: Ejemplo diferencia calidad (pito ibérico) | 42 |
| Ilustración 32: ejemplo distancias (pito ibérico)..... | 42 |

| | |
|---|----|
| Ilustración 33:ejemplos tranformaciones | 44 |
| Ilustración 34: Entrenamiento ResNet50 | 46 |
| Ilustración 35: Entrenamiento Inception_Resnet_V2 | 46 |
| Ilustración 36: Entrenamiento VGG16 | 46 |
| Ilustración 37: Entrenamiento MobileNet..... | 46 |
| Ilustración 38: Entrenamiento EficientNetB5 | 47 |
| Ilustración 39:Porcentaje de Val Accuracy y Test accuracy | 48 |
| Ilustración 40:Ejemplos base de datas | 48 |
| Ilustración 41:Entrenamiento Inception_Resnet_V2 | 49 |
| Ilustración 42:Entrenamiento EficientNetB5 | 49 |
| Ilustración 43: Prueba Tortola turca..... | 50 |
| Ilustración 44: Prueba Garcilla bueyera | 51 |
| Ilustración 45: Prueba Mirlo Común..... | 51 |
| Ilustración 46:Prueba verdecillo | 51 |
| Ilustración 47: Prueba misma especie | 52 |
| Ilustración 48: Prueba varias especies | 52 |

Índice Tablas

| | |
|--|----|
| Tabla 1: Hardware empleado en el proyecto..... | 34 |
| Tabla 2: Arquitecturas usadas | 43 |
| Tabla 3: Resultados de entrenamiento y validación tras 15 épocas | 45 |
| Tabla 4: Resultados Test_accuracy | 47 |
| Tabla 5 : Resultados de entrenamiento y validación..... | 49 |
| Tabla 6: Resultados Tes_accuracy | 50 |

Estructura

Este proyecto está estructurado de la siguiente forma:

- **Capítulo 1 ‘Introducción y motivación’:** Breve introducción al proyecto, la motivación que ha llevado a realizarlo y su principal objetivo.
- **Capítulo2 ‘Marco teórico’:** Explicación sobre todo los conceptos necesarios de la Inteligencia artificial y el *Deep Learning* para comprender el proyecto, se profundizará sobre las redes neuronales y métodos de entrenamiento.
- **Capítulo 3 ‘Implementación’:** Explicación del procedimiento que se va a seguir en la realización del clasificador de aves con *Deep Learning*, la selección de datos, las arquitecturas empleadas etc
- **Capítulo 4 ‘Experimentación’:** En este apartado se verán las pruebas realizadas para validar el algoritmo. En ellas se realizará una comparación entre las redes neuronales empleadas.
- **Capítulo 5 ‘Conclusiones y trabajos futuros’:** Para finalizar se expondrán las conclusiones obtenidas y se comentará la visión de futuro de este proyecto.

1. Introducción

En este proyecto hemos explorado el campo de la inteligencia artificial y su aplicación al mundo de la detección de imágenes automática, en este caso específico, en la detección de imágenes de aves

En este punto veremos desde la motivación que ha llevado realizar esta propuesta, la propuesta en sí y los objetivos marcados, el estado del arte y, por último, la estructura que va a tener el documento.

1.1 Motivación

Las Áreas Naturales Protegidas son lugares que deben gestionarse de forma sostenible para la conservación a largo plazo de su biodiversidad y la promoción de beneficios sociales directos e indirectos, estos se agrupan en cuatro grandes categorías: ambientales (relacionadas con el clima local y global) ecosistémico (calidad del agua y control de gases atmosféricos); social (recreación); y económico (desarrollo de actividades empresariales). Si bien estas áreas son especialmente vulnerables a las presiones antropogénicas como el Turismo o la Agricultura, se han previsto pocos procedimientos de digitalización para asegurar su desarrollo sostenible, lo cual es fundamental para una evaluación continua de la interacción entre los sistemas ecológicos y sociales para fomentar la biodiversidad, asegurando así al mismo tiempo la generación de riqueza y puestos de trabajo.

Por todo esto, en los últimos años se está buscando cambiar el modelo actual de gestión del turismo mediante el desarrollo de un modelo más centrado en la tecnología mediante enfoques de detección y modelado que permita una mejor integración de los datos, análisis y visualización. Proporcionando así una mejor información a la gente interesada, como gestores de turismo, políticos o investigadores.

Para conseguirlo, se está planteando diseñar sensores basados en visión artificial para el seguimiento, clasificación y censo de diferentes especies de aves. Nuevas arquitecturas IoT con topologías adaptativas basadas en mallas, utilizando protocolos de conectividad de baja potencia y largo alcance, podrían usarse para cubrir las zonas

deseadas. También se plantean procedimientos de gestión de datos sobre las condiciones de las áreas protegidas y un mejor *service layer* de gestión de datos ecológicos y turísticos, para facilitar la creación de productos y servicios que fomente la sostenibilidad de las Áreas Naturales Protegidas, teniendo en cuenta así todo el ciclo de vida de los datos: adquisición, preparación y consumo.

A parte de esta necesidad existente también hay una motivación personal en este proyecto, en mi etapa como estudiante no he recibido mucho conocimiento sobre inteligencia artificial y su aplicación a la detección de imágenes, por lo que veo una gran posibilidad de investigar y aprender sobre este tema tan importante actualmente.

1.2 Propuesta y objetivos

En los humedales de la provincia de Alicante existen diversas áreas de gran interés medioambiental que concentran poblaciones de aves de diversas especies. Existe por tanto una necesidad de establecer un censo de estas aves, de esta idea nace esta propuesta y para lograrlo se propone el uso de técnicas de *Deep Learning* para reconocerlas y clasificarlas según su especie en base a fotografías obtenidas en los lugares de interés.

Por ello, nacen los siguientes objetivos con la finalidad de realizar la propuesta.

- Aprender sobre el reconocimiento automático de imágenes con *Deep Learning*
- Investigar y estudiar cuales son los procesos más empleados por la comunidad científica para la clasificación de animales actualmente.
- Desarrollar y estandarizar un *dataset* completo para poder entrenar correctamente.
- Por último, realizar un clasificador imágenes de aves mediante *Deep Learning*, que sea capaz clasificar con el mayor acierto posible y así diferenciar entre cientos de clases de aves que se puedan avistarse en los parques naturales de la provincia de Alicante.

1.3 Estado del arte

En este apartado se realizará una pequeña contextualización del proyecto, para ello hará un breve repaso a la actualidad de reconocimiento de animales de aves y a las investigaciones que se hay de clasificación de aves u otros animales mediante redes neuronales.

Primero se comentará brevemente el laboratorio puntero a nivel mundial en la identificación de aves, seguido de esto se presentan las bases de datos públicas de donde se toman imágenes e información de aves. También se comentarán las principales investigaciones relacionadas, las arquitecturas que utilizan junto las fuentes de datos de estas.

1.3.1 TheCornellLab

El laboratorio de Ornitología de Cornell [7] perteneciente a la Universidad de Cornell en Ithaca, Nueva York, que se especializa en el estudio de aves y otros animales salvajes. Actualmente son una referencia mundial en el mundo de la ornitología debido a sus proyectos.

Los proyectos de TheCornellLab están financiados en su totalidad por subvenciones, patrocinios y donaciones. Algunos de los proyectos más famosos son:

- **EBird** [8]: Es el sitio web de observaciones de aves más famoso a nivel mundial, proporciona a científicos, investigadores y naturalistas aficionados, datos en tiempo real sobre la distribución y abundancia de aves.
- **Merlin Bird ID** [9]: Es una aplicación desarrollada por TheCornellLab para iOS y Android que te ayuda a identificar aves en tiempo real mediante fotos o sonidos.



Merlin Bird ID
From the Cornell Lab of Ornithology

Ilustración 1: Logo de Merlin Bird ID (<https://okdiario.com/tecnologia/merlin-bird-photo-id-2597550>)

1.3.2 iNaturalist

iNaturalist [10] es una iniciativa conjunta de la Academia de Ciencias de California y la National Geographic Society. Actualmente Es una de las webs de naturaleza más conocidas en el mundo. Al igual que en el caso de TheCornellLab. Tiene una amplia comunidad de científicos y naturalistas, los cuales registran y comparten sus observaciones. Estas son uso público por lo que se podrán extraer muchos datos necesarios.

iNaturalist



CALIFORNIA
ACADEMY OF
SCIENCES



NATIONAL
GEOGRAPHIC

Ilustración 2: iNaturalist (<https://colombia.inaturalist.org/>)

1.3.3 Arquitecturas de aprendizaje profundo para reconocimiento visual

Actualmente hay muchas maneras de realizar e implementar un clasificador de imágenes. Por ello de cara a ver cómo se va a afrontar este proyecto se ha estudiado la metodología que se ha empleado en otros trabajos e investigaciones.

Si buscamos investigaciones sobre aves recientes, en 2020 tenemos [1] sobre aves pequeñas, en este trabajo se recalca el potencial del uso de redes neuronales artificiales

profundas ya que te permite conseguir una mayor generalización en el modelo que con otras técnicas.

En [2] podemos ver como para hacer un clasificador de aves endémicas desarrollan un algoritmo basado en *Transfer Learning*. En cuanto a la arquitectura se emplea una *inception ResNet-v2* y para comparar resultados se emplearon también las siguientes redes convolucionales de última generación: *Inception-v3*, *Xception*, *ResNet101* y *MobileNetV2*.

Aunque [3][5] no sean investigaciones específicas de aves, la finalidad y las condiciones del estudio es muy similar al de este proyecto. Al igual que se ha comentado en el artículo anterior se emplea *Transfer Learning* para realizar los modelos con las siguientes arquitecturas de redes convolucionales: *AlexNet*, *VGGNet*, *GoogLeNet* y *Resnets*.

Como se puede ver en las investigaciones recientes relacionadas con la detección y clasificación de animales, se están empleando redes neuronales convolucionales y en especial el método más empleado es el *Transfer Learning* [2][3][5], por ello para realizar este proyecto se decidió usar redes preentrenadas en vez de crear una desde cero.

1.3.4 Fuentes de Datos

Los datos son una parte muy importante a la hora de hacer que un modelo predictivo funcione, sea efectivo y consiga generalizar. Para conseguir esto los datos deben ser precisos, válidos y reales.

En [4] se puede encontrar información sobre data relacionada con especies, hábitats y localizaciones. Esto nos será útil a la hora de tener que recopilar nuestros propios datos, algunos de los portales de datos que se nombran son:

- GBIF – Global Biodiversity Information Facility
- iNaturalist
- The National Biodiversity Network UK
- EVA – European Vegetation Archive
- Copernicus

También se recogen algunas de las técnicas más empleadas para la recopilación de datos en animales, y como hemos podido comprobar en el resto de las investigaciones [1][3][5], la aportación humana y el uso de sensores es lo más extendido actualmente.

Las cámaras con sensor de movimiento, también llamadas cámaras de fototrampeo, ofrecen en hábitats naturales la oportunidad de recopilar de manera económica y discreta grandes cantidades de datos sobre animales en la naturaleza. Su principal inconveniente es el gran costo humano para analizar cada imagen. Por ello es necesario el uso de inteligencia artificial para realizar este trabajo de un forma más eficaz y sencilla.

Para iniciarse en el mundo del *Deep Learning* y empezar a trabajar en nuestro modelo se decidió usar una base de datos externa. Tras realizar una búsqueda se prefirió usar la base de datos de Nabirds.

Este conjunto de datos que ofrece *TheCornellLab*, está dividido en *Entrenamiento*, *Test* y *Validación*. Contiene 48.562 imágenes de aves divididas en 400 clases, en su mayoría, por no decir en su totalidad, de Norte América.

Todas las imágenes están etiquetadas con la especie a la que pertenecen cada ave. Las fotografías están tomadas en diferentes puntos de vista (en tierra, volando...)



Ilustración 3: Camera trap (<https://malleedesign.com.au/camera-trap-basics-for-bird-spotters/>)

2 Marco teórico

En este capítulo se realiza una introducción a las tecnologías, arquitecturas, herramientas y utilidades que han servido como base en el desarrollo del proyecto

2.1 Inteligencia artificial

La **inteligencia artificial (IA)** [11] es la base a partir de la cual se imitan, mediante algoritmos creados en un entorno dinámico de programación, los procesos de inteligencia humana, o dicho de una manera más sencilla de entender, la IA consiste en intentar que los ordenadores piensen y actúen como los humanos.

En concreto este proyecto se va a usar la rama de *Machine Learning*, el *Deep Learning* y *Big Data*.

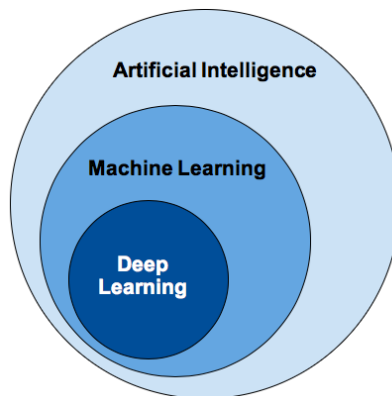


Ilustración 4: Diagrama de Venn de la Inteligencia artificial(<https://www.google.com/imgres?imgurl=https%3A%2F%2Fwww.sumologic.com>)

2.2 Machine Learning

El **Aprendizaje automático o Machine Learning (ML)** [12] es una rama de la IA y la informática que se centra en el uso de datos y algoritmos con la finalidad de enseñar a los ordenadores a aprender como lo hacen los seres humanos, a través de la experiencia. Lo que se busca es generar a partir de unas muestras de datos un modelo que sea capaz de generalizar comportamientos para un conjunto de datos mayor al usado inicialmente.

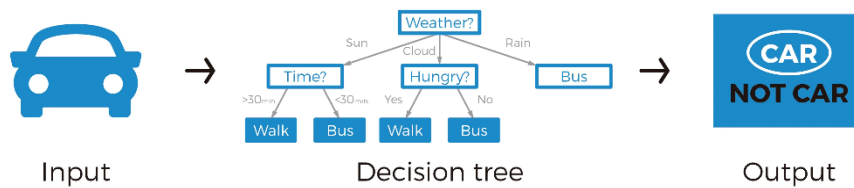


Ilustración 5: Esquema ejemplo Machine Learning (<https://victoryepes.blogs.upv.es/2020/09/15/el-aprendizaje-profundo-deep-learning-en-la-optimizacion-de-estructuras/>)

El *Machine Learning* se puede dividir en tres tipos dependiendo de forma de aprender:

- **Aprendizaje supervisado:** el algoritmo “aprende” de los datos introducidos por una persona. Se necesita intervención humana para clasificar, etiquetar e introducir los datos en el algoritmo, el cual ya genera los datos de salida

Los datos introducidos pueden ser de tipo **clasificación** si clasifican un objeto dentro de diversas clases o de **regresión** si lo que están haciendo es predecir un valor numérico.

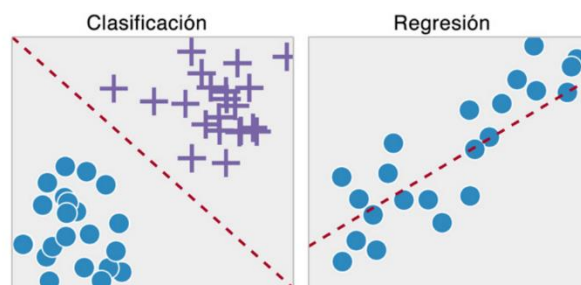


Ilustración 6: Ejemplo Clasificación y Regresión (<https://medium.com/iwannabedatadriven/machine-learning-modelos-de-regresi%C3%B3n-i-d293ae235e9a>)

- **Aprendizaje no supervisado:** en este caso no existe la intervención humana, los algoritmos aprenden mediante datos con elementos no etiquetados buscando patrones o relaciones entre ellos.

Hay dos tipos de *Machine Learning* no supervisado:

- **Clustering:** los datos de salida se clasifican en grupos
- **Asociación:** descubre reglas dentro de un conjunto de datos.

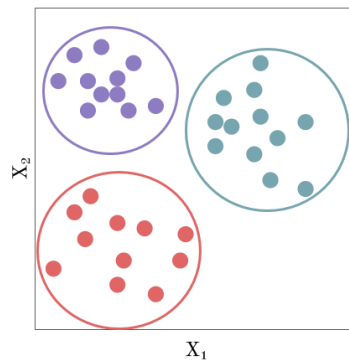


Ilustración 7: Ejemplo de Clustering (https://favpng.com/png_view/robotics-machine-learning-unsupervised-learning-deep-learning-png/RrkZKc9X)

- **Aprendizaje por refuerzo:** este método es menos empleado que los anteriores, está inspirado en la psicología conductista, el algoritmo aprende solo en base a recompensas y penalizaciones.



Ilustración 8: Esquema ejemplo aprendizaje por refuerzo (<https://medium.com/soldai/tipos-de-aprendizaje-autom%C3%A1tico-6413e3c615e2>)

2.3 Deep Learning

El **Aprendizaje Profundo o Deep Learning** [13] es un categoría de *Machine Learning* que permite que las computadoras extraigan automáticamente múltiples niveles de abstracción de los datos sin procesar. Esto se lleva a cabo mediante el uso de redes neuronales utilizadas en diversas estructuras con el objetivo de realizar un aprendizaje automático a través de una serie de capas.

Como se ha comentado, el *Deep Learning* realiza estas tareas a través de una red neuronal artificial compuesta por múltiples capas definidas en el proceso de creación de la red. En primer lugar, siempre tenemos la capa inicial de la red, posteriormente, se envía la información a la siguiente capa. En cada capa la información se vuelve más compleja pero la red es capaz de modelar relaciones más complejas. Estas relaciones vienen determinadas por los pesos de la red, que son el objeto del aprendizaje.

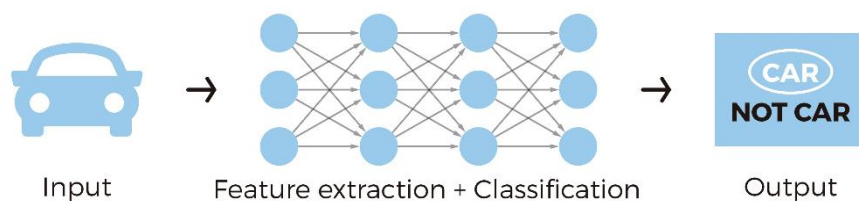


Ilustración 9: Esquema ejemplo Machine Learning (<https://www.crehana.com/blog/desarrollo-web/que-deep-learning/>)

2.4 Redes neuronales artificiales

El funcionamiento de las redes neuronales artificiales está inspirado en la organización neuronal del cerebro humano. Se trata de un modelo computacional que consiste en un conjunto de neuronas artificiales conectadas entre sí mediante enlaces para transmitirse señales. De esta manera, entendemos las redes neuronales como un sistema en el cual tenemos unos datos de entrada, los cuales pasan a través de las capas internas en donde se hacen transformaciones, para que una vez la información llegue a la salida, el resultado sea el deseado.

Como se puede observar en la *Ilustración 10*, están compuestas por tres capas, una de entrada, una oculta y otra de salida.

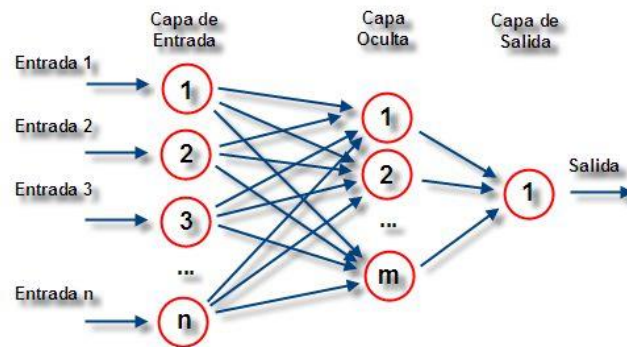


Ilustración 10: Esquema red neuronal artificial ([Perceptrón multicapa - Wikipedia, la enciclopedia libre](#))

2.5 Red Neuronal Convolutiva (CNN)

Las Redes neuronales convolucionales [14] son un tipo de redes neuronales artificiales donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria de un cerebro. Este tipo de red está formada por múltiples capas de filtros convolucionales, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones.

Al ser una red empleada para clasificación, al principio tiene una fase de extracción de características, compuesta de neuronas convolucionales, luego hay una reducción por muestreo y al final tendremos neuronas de perceptrón más sencillas para realizar la clasificación final sobre las características extraídas. En definitiva, las CNN no son más que redes neuronales en las cuales se usa la operación matemática de convolución en una de sus capas sobre la matriz de píxeles. Las redes convolucionales enfocadas a la clasificación siguen la siguiente estructura de capas.

- **Capa convolutiva:** Filtrado de imágenes.
- **Capa de reducción (Pooling):** Reducción de parámetros.
- **Capa clasificadora Fully Connected** Clasificación del resultado final.

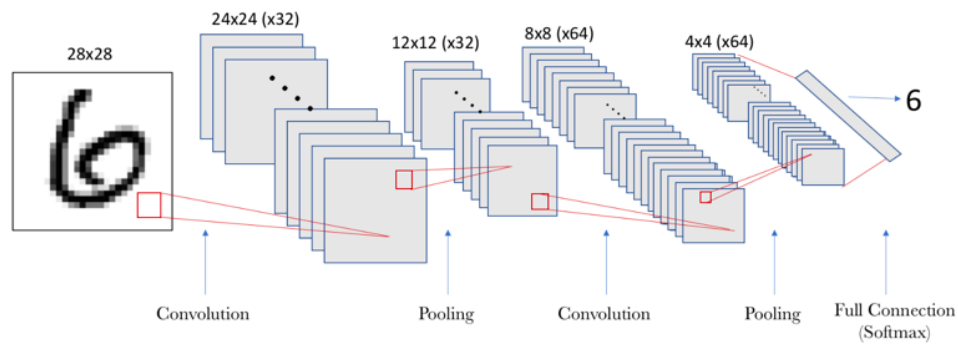


Ilustración 11: Esquema CNN (<https://torres.ai/deep-learning-inteligencia-artificial-keras/>)

2.5.1 Capa convolucional

Esta capa consistirá en seleccionar un grupo de píxeles cercanos de la imagen y realizar un producto escalar con un filtro *Kernel* elegido que se irá desplazando por todos los píxeles de la imagen y se obtendrá una nueva matriz con el resultado, a esta matriz se le conoce como matriz de activación. Los píxeles de esta nueva matriz de salida son el resultado de la combinación lineal de los píxeles de imagen de entrada.

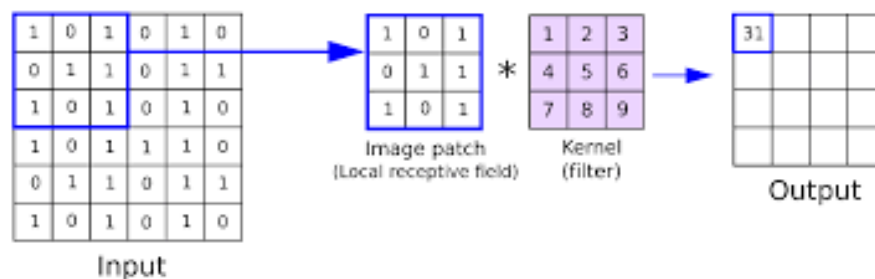


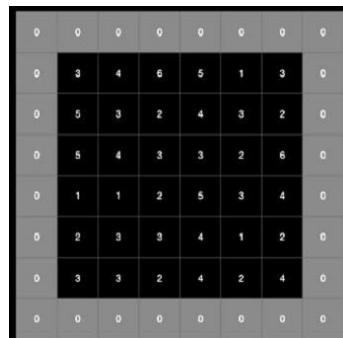
Ilustración 12: Proceso capa convolucional ([Convolutional Neural Networks: La Teoría explicada en Español / Aprende Machine Learning](#))

Para que se entienda mejor, se van a explicar los parámetros más importantes de esta capa

- **Kernel:** en las redes convolucionales se considera como el filtro que se aplica a una imagen para extraer ciertas características importantes o patrones de esta. Entre las características importantes para lo que sirve el *kernel* son detectar bordes, enfoque, desenfoque, entre otros. Esto se logra al realizar la convolución

entre la imagen y el *kernel*. Cuando la imagen es en color, se usarán 3 *kernels* (RGB) que se sumarán para obtener la imagen de salida

- **Padding:** es una operación que se usa en redes convolucionales, se aplica agregando píxeles de valor cero alrededor de la imagen original. Esto se hace para que al realizar la convolución la imagen resultante sea de igual tamaño que la imagen original y también se emplea cuando se tiene información relevante en las esquinas de la imagen, se aplica el *padding* para tener la información más relevante cerca del centro.



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 4 | 6 | 5 | 1 | 3 | 0 | 0 | 0 |
| 0 | 5 | 3 | 2 | 4 | 3 | 2 | 0 | 0 | 0 |
| 0 | 5 | 4 | 3 | 3 | 2 | 6 | 0 | 0 | 0 |
| 0 | 1 | 1 | 2 | 5 | 3 | 4 | 0 | 0 | 0 |
| 0 | 2 | 3 | 3 | 4 | 1 | 2 | 0 | 0 | 0 |
| 0 | 3 | 3 | 2 | 4 | 2 | 4 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Ilustración 13: Ejemplo de Padding ([Intro a las redes neuronales convolucionales / by Bootcamp AI / Medium](#))

2.5.2 Pooling

Esta capa se emplea para submuestrear una representación de entrada (imagen, matriz de salida de capa escondida, etc.) reduciendo así la resolución de las imágenes. Es útil porque reduce el coste de cálculo, reduciendo el número de parámetros que tiene que aprender y proporciona una invariancia por pequeñas translaciones. Existen varios métodos:

- **Max pooling:** Se elige el máximo valor de una ventana de la matriz activación previamente obtenida.
- **Average pooling:** en este método se escoge el valor de la media de la ventana.

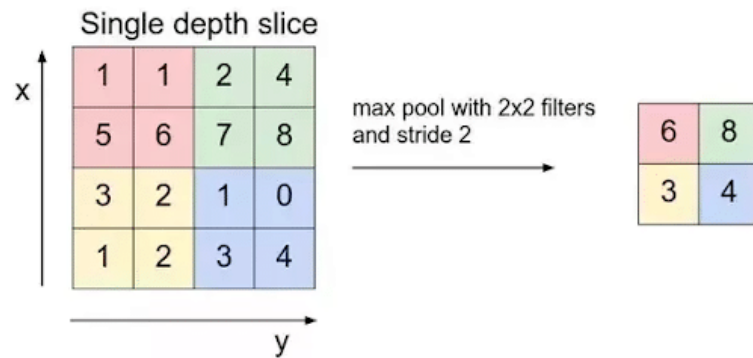


Ilustración 14: procesamiento del Max Pooling (<https://stackoverflow.com/questions/48549670/pooling-vs-pooling-over-time>)

2.5.3 Capas de activación

Cada una de las capas que conforman la red neuronal tienen una función de activación que permitirá reconstruir o predecir. La función de activación devuelve una salida que será generada por la neurona dada una entrada o conjunto de entradas. A continuación, veremos algunas funciones de activación:

2.5.3.1 Sigmoide

La función sigmoide transforma los valores introducidos a una escala (0,1), donde los valores altos tienden de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0

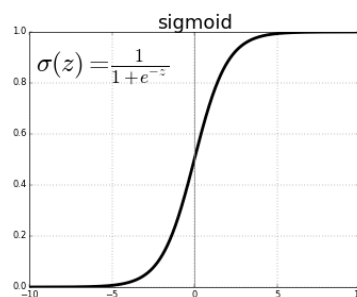


Ilustración 15: Función de activación Sigmoide([Redes Neuronales \(ml4a.github.io\)](https://github.com/josiah Davis/ml4a))

2.5.3.2 Tangente hiperbólica

La función tangente hiperbólica transforma los valores introducidos a una escala (-1,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a -1.

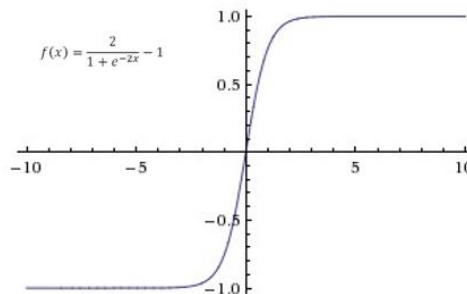


Ilustración 16: Función de activación Tangente hiperbólico([Redes Neuronales \(ml4a.github.io\)](https://ml4a.github.io))

2.5.3.3 ReLu:

Esa capa sustituye todos los valores negativos recibidos en la entrada por ceros. El interés de esas capas de activación es hacer que el modelo sea no lineal y por tanto, más complejo.

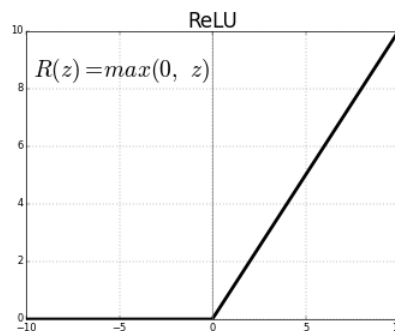


Ilustración 17: Función de activación ReLu([Redes Neuronales \(ml4a.github.io\)](https://ml4a.github.io))

2.5.3.4 Softmax

La función *Softmax* transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas de 1.

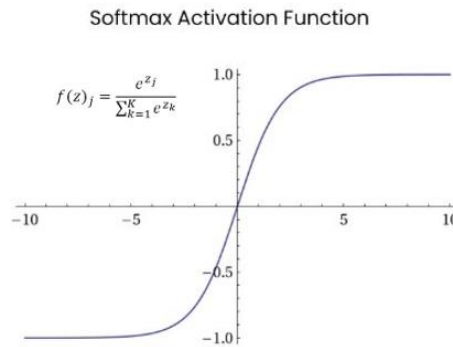


Ilustración 18: Función de activación Softmax([Redes Neuronales \(ml4a.github.io\)](https://ml4a.github.io/))

2.5.4 Capa Fully Connected:

Estas capas se colocan al final de la arquitectura de las redes convolucionales y se conectan por completo a todas las neuronas de salida. Reciben un vector de entrada al que le aplica una combinación lineal y una función de activación con la finalidad de clasificar la imagen.

2.6 Transfer Learning

Es una técnica que permite a un algoritmo de *Machine Learning* mejorar sus capacidades de aprendizaje en un *dataset*, exponiéndolo previamente a uno totalmente diferente. Es decir, se hace uso de una red pre entrenada, así no necesitaremos hacer uso de una gran cantidad de datos ni nos tomará tanto tiempo el entrenar nuestra red.

En el caso de *computer vision* hay muchos modelos previamente entrenados, habitualmente con el conjunto de datos *ImageNet*, que está disponible para usarse y poder crear modelos potentes usando menos datos que si se entrenara desde 0.

Especialmente cuando vamos a hacer una clasificación de imágenes suele ser

recomendable hacer uso de modelos ya previamente entrenados, aunque si el modelo está preentrenado para un problema que no está relacionado con el modelo que deseas, es aconsejable hacerlo, pues las capas interiores que han aprendido por ejemplo a detectar los bordes y otros patrones son de gran utilidad.

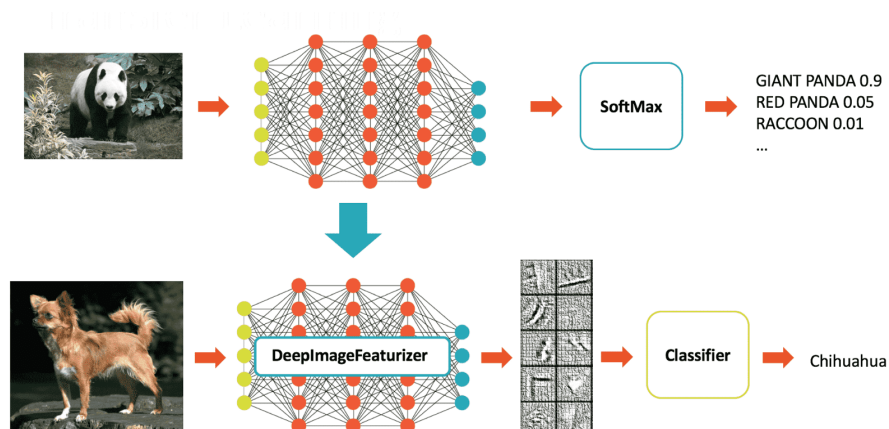


Ilustración 19: Esquema ejemplo de Transfer Learning (<https://blog.superannotate.com/speed-up-labeling-process-using-transfer-learning/>)

2.7 Arquitecturas

En este apartado se van a explicar las principales arquitecturas de redes neuronales convolucionales que se van a emplear en este proyecto.

2.7.1 VGGNet

Es una arquitectura [15] del laboratorio Visual Geometry Group en Oxford. Su modelo fue desarrollado y demostrado para el reto ILSVRC, en este caso, la versión ILSVRC-2014 del desafío.

La entrada a la red es una imagen de dimensiones (224, 224, 3). Las dos primeras capas tienen 64 canales de un tamaño de filtro de 3x3 y el mismo *padding*. Luego, después de una capa *max-pooling* (2, 2), dos capas tienen capas de convolución de 128 tamaños de filtro y tamaño de filtro (3, 3). A esto le sigue una capa *Max-pool* de agrupación máxima (2, 2) que es la misma que la capa anterior.

Las siguientes 2 capas de convolución tienen 256 filtros de un tamaño de filtro (3, 3). Después de eso, hay 2 conjuntos de 3 capas de convolución y una capa de grupo máximo. Cada uno tiene 512 filtros de tamaño (3, 3) con el mismo patrón.

Luego, esta imagen se pasa a la pila de dos capas de convolución. En algunas de las capas, también usa 1x1 píxel que se usa para manipular la cantidad de canales de entrada. Hay un patrón de 1 píxel (mismo patrón) realizado después de cada capa de convolución para evitar la característica espacial de la imagen.

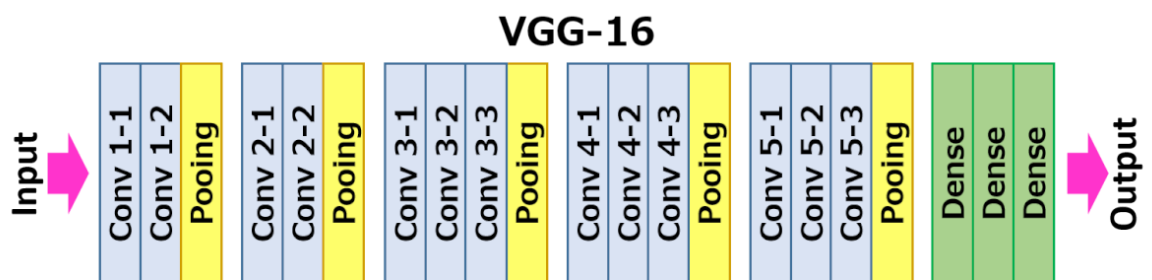


Ilustración 20: arquitectura vgg16 <https://neurohive.io/en/popular-networks/vgg16/>

2.7.2 Resnet50

ResNet50 [16] es una variante del modelo *ResNet* que tiene 48 capas de convolución junto con 1 *MaxPool*, 1 capa de promedio y activación *ReLU*. Tiene operaciones de $3,8 \times 10^9$ puntos flotantes.

Está inspirada en VGG en la que luego se agrega la conexión de acceso directo. Estas conexiones de acceso directo luego convierten la arquitectura en una red residual.

Como se puede observar en la *Ilustración 21* consiste en el módulo principal, cuatro módulos residuales y una capa de red neuronal completamente conectada.

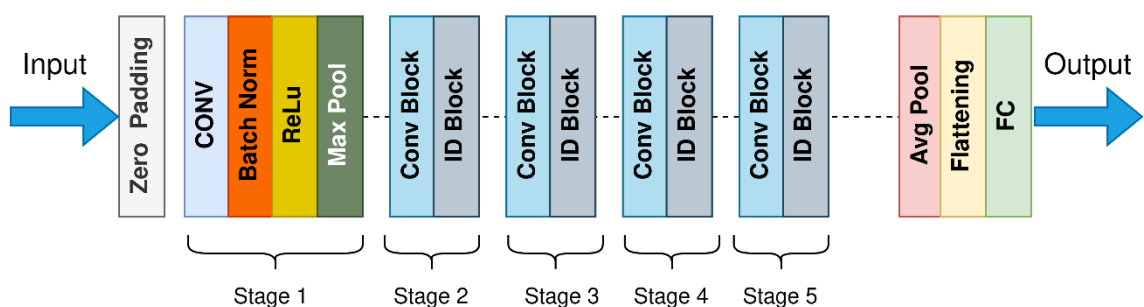


Ilustración 21: Arquitectura Resnet50 <https://commons.wikimedia.org/wiki/File:ResNet50.png>

2.7.3 Inception ResNet-v2

Esta arquitectura [17] nació de una investigación tras el éxito de las residual Network (ResNet), en ella exploraron la posibilidad de combinar Inception con Resnets.

Tanto la red *Inception* como la *Residual* son arquitecturas SOTA, que han mostrado muy buen desempeño con un costo computacional relativamente bajo. *Inception-ResNet V2*-combina las dos arquitecturas para aumentar aún más el rendimiento.

Como se puede observar en la *Ilustración 22* la imagen de entrada es de 299x299x3.

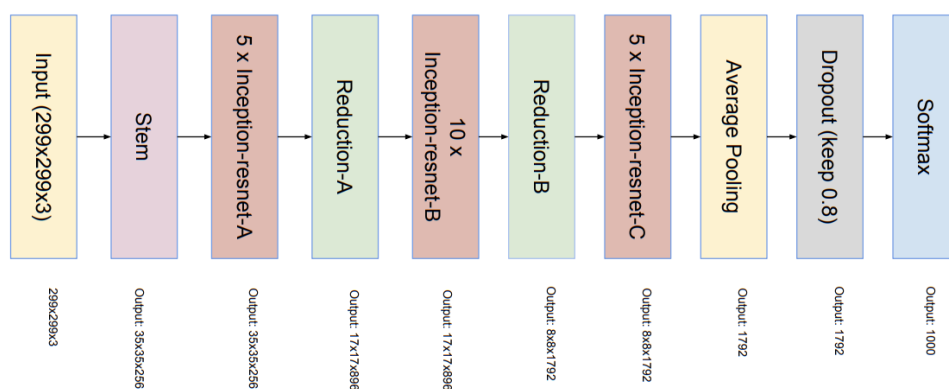


Ilustración 22: Arquitectura inception-ResNet-v2 <https://paperswithcode.com/method/inception-resnet-v2>

2.7.4 MobileNet

El Laboratorio de Redes Móviles (Mobilenet) [18] es una unidad de investigación perteneciente al Instituto de Telecomunicaciones (TELMA) de la Universidad de Málaga. El grupo Mobilenet está especializado en investigación y desarrollo relacionado con las redes de generación futura.

Según los creadores, MobileNet es una arquitectura *CNN* computacionalmente eficiente diseñada específicamente para dispositivos móviles con una potencia informática muy limitada. Se puede utilizar para diferentes aplicaciones, entre ellas: detección de objetos,

Están construidas sobre capas de convolución separables profundas . Cada capa de convolución separable en profundidad consiste en una convolución profunda y una convolución en puntos. Contando las convoluciones en profundidad y en puntos como

capas separadas, una MobileNet tiene 28 capas. Una MobileNet estándar tiene 4.2 millones de parámetros que pueden ser más se reduce ajustando el hiperparámetro del multiplicador de anchura de forma adecuada.

El tamaño de la imagen de entrada es $224 \times 224 \times 3$ como se puede ver en arquitectura de la *Ilustración 23*.

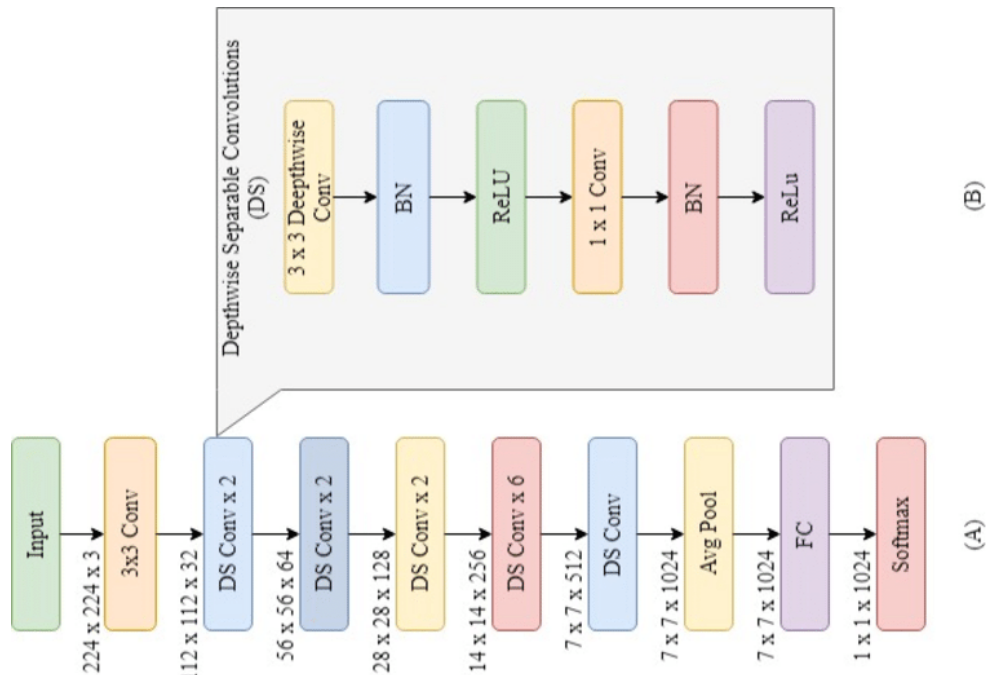


Ilustración 23: Arquitectura MobileNet https://www.researchgate.net/figure/Illustration-of-the-MobileNet-architecture-A-The-overall-MobileNet-architecture-and_fig3_340470168

2.7.5 EficientNet

Es un método [19] de escalado de modelos novedoso que utiliza un coeficiente compuesto simple pero altamente efectivo. A diferencia de los enfoques convencionales que escalan arbitrariamente las dimensiones de la red, como el ancho, la profundidad y la resolución, este método escala uniformemente cada dimensión con un conjunto fijo de coeficientes de escala.

La eficacia del escalado del modelo también depende en gran medida de la red de referencia. Por lo tanto, para mejorar aún más el rendimiento, se desarrolló una nueva red de referencia al realizar una búsqueda de arquitectura neuronal utilizando el marco *AutoML MNAS*, que optimiza tanto la precisión como la eficiencia (*FLOPS*). La arquitectura resultante utiliza convolución de cuello de botella invertida (*MBConv*),

similar a *MobileNetV2* y *MnasNet* , pero es un poco más grande debido a un mayor presupuesto de *FLOP*.

La arquitectura de esta red es simple y limpia, lo que facilita su escalado y generalización. El tamaño de la imagen de entrada es $224 \times 224 \times 3$.

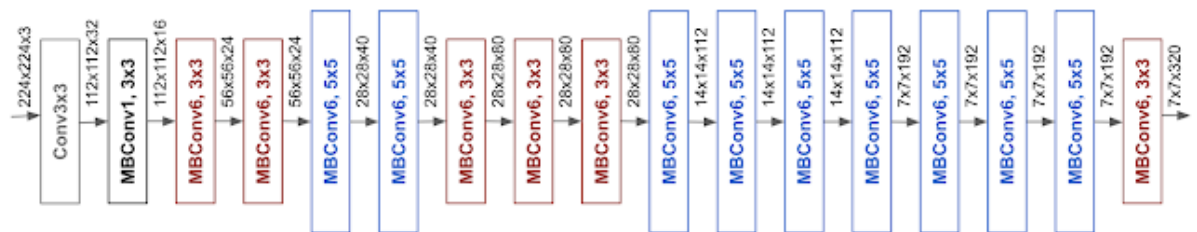


Ilustración 24:Arquitectura EfficientNet <https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html>

2.8 Overfitting

El *overfitting* es uno de los problemas que nos podemos encontrar a la hora de entrenar un modelo, esto ocurre cuando el modelo considera como validado, solo los datos que se han usado para entrenar el modelo, sin reconocer otros datos diferentes que debería clasificar sin problemas.

Usar *Transfer Learning* es una forma de intentar que se produzca este problema por falta de datos de entrada.

La principal forma de detectar el *overfitting* es fijarse si al entrenar nuestro modelo detectamos que la curva de error del training difiere mucho de la de *test*.

Existen algunas formas de evitar el *overfitting*:

- **Entrenamiento con más datos:** puede ayudar a identificar mejor la señal, pero también puede significar que aumente el ruido.
- **Detención anticipada:** significa detener el proceso de entrenamiento antes de que el modelo pase el punto donde comienza a producirse el *overfitting*.
- **Ensamblado (Ensemble):** Esta técnica básicamente combina predicciones de diferentes modelos. Con el *bagging* usa modelos complejos e intenta suavizar sus predicciones mientras que el impulso usa modelos base simples e intenta aumentar su complejidad agregada.

- **Parada anticipada (*Early stopper*):** cuando se entrena por iteraciones puedes medir qué tan bien se desempeña cada iteración del modelo. Con el paso de las iteraciones el modelo mejora, pero llega un momento que la capacidad para generalizar del modelo se debilita, esta técnica pararía el proceso antes de que esto pase.
- **Validación cruzada:** la idea usar tus datos de entrenamiento inicial para generar múltiples divisiones de prueba y utilizar estas divisiones para ajustar tu modelo.

2.9 Tecnologías

2.9.1 Software

En este apartado se analizan las principales herramientas o softwares que se han empleado en la realización de este proyecto.

2.9.1.1 Python:

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código.

El código Python es conciso y legible incluso para los nuevos desarrolladores, lo que es beneficioso para los proyectos de aprendizaje automático y profundo. Debido a su sintaxis simple, el desarrollo de aplicaciones con Python es rápido en comparación con muchos lenguajes de programación. Además, permite al desarrollador probar algoritmos sin implementarlos.

El lenguaje Python viene con muchas bibliotecas y marcos que facilitan la codificación. Esto también ahorra una cantidad significativa de tiempo. Las librerías que se han usado en este proyecto son las siguientes:

- **Numpy** [20]: es una librería Python *open source* especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos. Permite crear vectores y matrices de grandes dimensiones.
- **Pandas** [21]: es una librería especializada en el manejo y análisis de datos. Define

nuevas estructuras de datos en los vectores de la librería Numpy pero con nuevas funcionalidades

Ofrece métodos para reordenar, dividir y combinar conjuntos de datos y nos permite acceder a los datos mediante índices o nombres para filas y columnas de una manera muy eficiente.

- **Matplotlib** [22]: es una librería Python para crear gráficos estáticos, dinámicos e interactivos en dos dimensiones
- **Sklearn** [23]: esta librería es conjunto de rutinas escritas en Python, contiene muchas herramientas eficientes para aprendizaje automático y modelado estadístico, incluyendo clasificación, regresión, agrupación, y reducción de dimensionalidad. Está basada en NumPy, SciPy y matplotlib, de forma que es fácil reaprovechar el código que use estas librerías y se va a usar para realizar la matriz de confusión.

2.9.1.2 TensorFlow

Es una plataforma [24] de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite que los investigadores innoven con el aprendizaje automático. Se puede utilizar junto una API de alto nivel como Keras.

2.9.1.3 Keras

Es una API [25] de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow. Está diseñada para la experimentación con redes convolucionales y recurrentes. Se puede ejecutar en CPU y GPU.

2.9.1.4 Jupiter Notebook

Debido a que Keras [26] es básicamente una librería de Python, requerimos hacer un uso completo del intérprete de Python. Jupyter es un entorno de desarrollo muy extendido que permite sacar partido a la interactividad de Python y, a la vez, proporciona

una gran versatilidad para compartir parte de códigos con anotaciones a través de la comodidad e independencia de plataforma que ofrece un navegador web.

2.9.2 Hardware

El equipo que se ha usado para realizar el proyecto cuenta con las siguientes características.

| | |
|-------------------|------------------|
| Sistema operativo | Windows 10 |
| CPU | AMD Ryzen 5 3600 |
| RAM | 16GB DDR4 |
| GPU | NVIDIA GTX 2060 |

Tabla 1: Hardware empleado en el proyecto

3 Implementación

En este apartado se describen los modelos y arquitecturas desarrolladas en este proyecto, empezando por la generación de una base de datos al modelo y siguiendo con el modelo predictivo de este trabajo.

3.1 Planteamiento inicial

En el apartado de estado del arte se han visto las soluciones que da la comunidad científica al problema que queremos afrontar en este proyecto y casi todos coincidían en que el *Transfer Learning* era la mejor opción para conseguir un modelo eficaz.

Se tendrá que tomar una decisión sobre que arquitectura de red neuronal convolucional de las que han usado se emplea, con el objetivo de entrenarlos y verificar el rendimiento de cada uno de ellos.

También sabemos que para poder entrenar vamos a necesitar datos, ya que los datos son la materia prima que alimenta los algoritmos de *Deep Learning* y es muy importante obtener la mayor cantidad de datos posible y tenerlo bien etiquetados. Para ello se va a necesitar de una base de datos de aves autóctonas de la provincia de Alicante.

Como no existe ninguna base de datos con las características que necesitamos para nuestro estudio, hará falta recopilar datos y estandarizarlos para generar un *dataset* que podamos usar para realizar nuestro proyecto.

3.2 Obtención de los datos

Como se ha comentado anteriormente se va a tener que generar un *dataset* específico para este proyecto, lo ideal hubiera sido que ya existiera una base de datos que cumpliera con las necesidades de este estudio. Debido al gran trabajo que lleva realizar esto se decidió trabajar un número reducido de aves.

- **Varias aves en una misma foto:** Para asegurar que la red entrena correctamente se ha dejado solo un ave en una foto, tanto si son o no de la misma especie. A la izquierda se puede ver varios individuos de una especie y a la derecha varias especies juntas



Ilustración 26: ejemplos varias aves en una foto

- **Multitudes:** entre las imágenes hay algunas tomadas zonas en las que no se pueden distinguir las aves, como se ve en la *Ilustración 27*. Estas imágenes no se usaron en este trabajo.



Ilustración 27: ejemplo de multitud de aves

- **Plumas y huevos:** al ser imágenes para estudios científicos en algunos casos también hay fotos de plumas y huevos de las aves.



Ilustración 28: Ejemplo pluma y huevo

- **Aves muertas:** entre los datos de aves hay imágenes de aves muertas, estas imágenes no se han incluido.

3.2.3 Especies de estudio

A la hora de seleccionar las especies de estudio se buscó en eBird cuales eran las aves más vistas de la zona, a parte se tuvieron en cuenta otros factores, como que hubiera suficientes imágenes de esa especie para que no quedara desbalanceada y, por último, se le dieron prioridad a todas las aves que capto la cámara de fototrampeo y de las cuales ya teníamos información. Teniendo en cuenta todo esto, las especies de estudio son:

| | Nombre |
|----|----------------------|
| 1 | Serín Verdecillo |
| 2 | Tórtola Turca |
| 3 | Mirlo Común |
| 4 | Gorrión Común |
| 5 | Lavandera blanca |
| 6 | Paloma torcaz |
| 7 | Garcilla bueyera |
| 8 | Andarrios Grande |
| 9 | Archibebe |
| 10 | Pito real |
| 11 | Carbonero Común |
| 12 | Jilguero Europeo |
| 13 | Alcaraván |
| 14 | Abubilla |
| 15 | Perdiz roja |
| 16 | Estornino Negros |
| 17 | Pato cuchara Común |
| 18 | Golondrina Común |
| 19 | Focha Común |
| 20 | Garcilla Bueyera |
| 21 | Tarro blanco |
| 22 | Avoceta Común |
| 23 | Flamenco Común |
| 24 | Gaviota Patiamarilla |
| 25 | Gaviota Audouin |

3.3 Análisis de los datos

Los datos con los que contamos para entrenar y validar el modelo están compuestos por 9.5k imágenes en total. Todas las imágenes están etiquetadas con la especie a la que pertenece cada ave, y así poder cumplir con el objetivo de realizar el clasificador.

A pesar de tener todas las imágenes bien etiquetadas el *dataset* presenta una serie de características y dificultades que pueden generar un problema a la hora de entrenar y validar el modelo. Para que esto suceda en la menor escala posible se van a explicar los problemas y ver sus posibles soluciones.

3.3.1 Clases desbalanceadas

Uno de los principales problemas que tiene la base de datos generada es que la cantidad de ejemplos con los que contamos de cada clase es diferente entre sí. Aún seleccionando especies con bastantes ejemplos, sigue habiendo clases con más imágenes que otras.

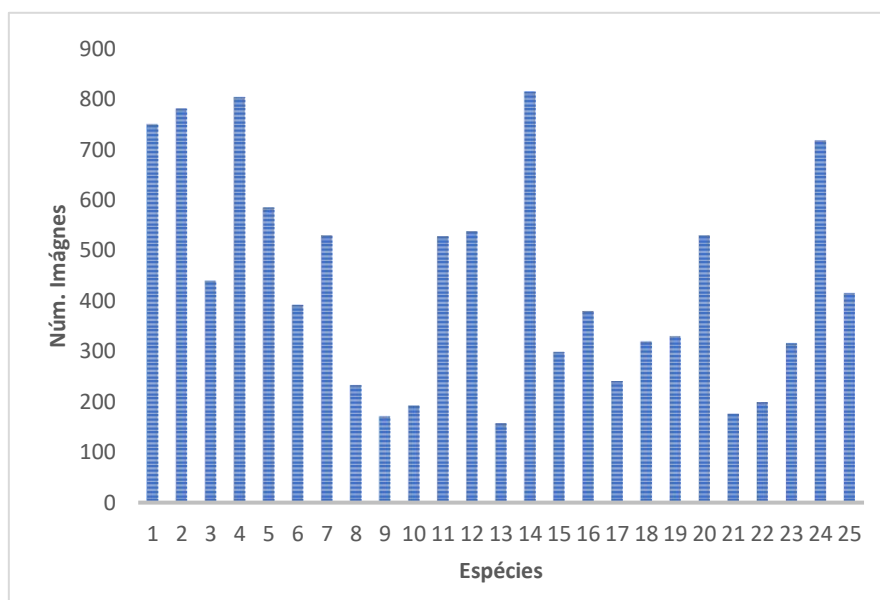


Ilustración 29: Gráfico especies desbalanceadas

3.3.2 Similitud entre imágenes

Otro problema que nos encontramos es la similitud entre algunas de las aves, esto puede ser debido a que varias aves pueden ser de la misma familia o simplemente porque se parezcan entre sí visualmente.

Por ejemplo, podemos ver cómo hay una gran similitud entre estos dos tipos de gaviotas, Audouin y Patiamarilla. Por eso en este tipo de casos es muy útil tener una gran cantidad de datos, para que el algoritmo pueda aprender esas pequeñas características que les diferencia. En este caso la diferencia es el color del pico.



Ilustración 30: ejemplo similitud (Gaviota de Audouin y Gaviota Patiamarilla)

3.3.3 Diferencia de calidad o distancia

Las fotografías han sido tomadas por usuarios por la comunidad por lo que puede existir diferencia entre las fotografías de una misma especie y también con respecto al resto.

Como se puede observar en el siguiente ejemplo tenemos fotografía de alta calidad y también existen otras más borrosas en la que el elemento se ve parcialmente. Aunque a la red neuronal le cueste más aprender de esta manera, es necesario tener una gran diversidad de casos para que en un futuro pueda generalizar.



Ilustración 31: Ejemplo diferencia calidad (pito ibérico)

Al igual que la calidad es diferente, la distancia a la que son tomadas la fotografías también varía lo que hará más difícil reconocer al ave. Como se ha explicado anteriormente esto será muy útil a la hora de generalizar.



Ilustración 32: ejemplo distancias (pito ibérico)

3.4 Modelo

Tras analizar el conjunto de datos se va a explicar el entramiento del modelo, teniendo en cuenta los parámetros y técnicas que se han empleado para realizarlo.

Lo primero es comentar que se convirtieron todas las imágenes al mismo formato (“.PNG”) mediante un script de Python y se dividió nuestra base de datos en tres subconjuntos:

- Train (70%).
- Val (20%).
- Test (10%).

El conjunto de datos, *Nabirds*, que se va a usar en la elección de las arquitecturas candidatas viene separado también en estos tres subconjuntos.

3.4.1 Arquitecturas usadas

Como ya se ha comentado en este proyecto se han empleado arquitecturas de CNN ya existentes. El objetivo va a ser probar primero su funcionamiento con los datos de *Nabirds* y ver que redes tienen mejores resultados, para posteriormente entrenar las mejores con nuestros datos y verificar su rendimiento.

Los modelos han sido preentrenados con *ImageNet*, una base de datos de 14 millones de imágenes y más de 20.000 categorías diferentes, este conjunto de datos se usa como referente en el reconocimiento de imágenes. Las arquitecturas que se han empleado son las siguientes:

| Nombre | Número de parámetros |
|----------------------------|----------------------|
| Vgg16 | 14,714,688 |
| Resnet50 | 25,636,712 |
| Inception_Resnet_V2 | 50,000,00 |
| Mobilenet | 13,000,00 |
| EfficientNetB5 | 30,000,000 |

Tabla 2: Arquitecturas usadas

3.4.2 Aumentado de datos

Es una técnica realizada con la finalidad de aumentar el número de imágenes con el que se va a entrenar la red. Como se ha comentado en puntos anteriores, existe un desequilibrio entre el número de muestras de cada especie, con esta técnica podremos conseguir más muestras de las clases que tienen menos.

Para aumentar el número de imágenes, se suelen usar transformaciones como rotaciones, variaciones de brillo y contraste, ampliaciones o añadir ruido entre otras técnicas. En concreto, las variaciones que se van a realizar son:

- Rotación (10 grados)
- Rescalado (255x255)
- Zoom (10%)
- Variaciones en el ancho y el alto (10%)

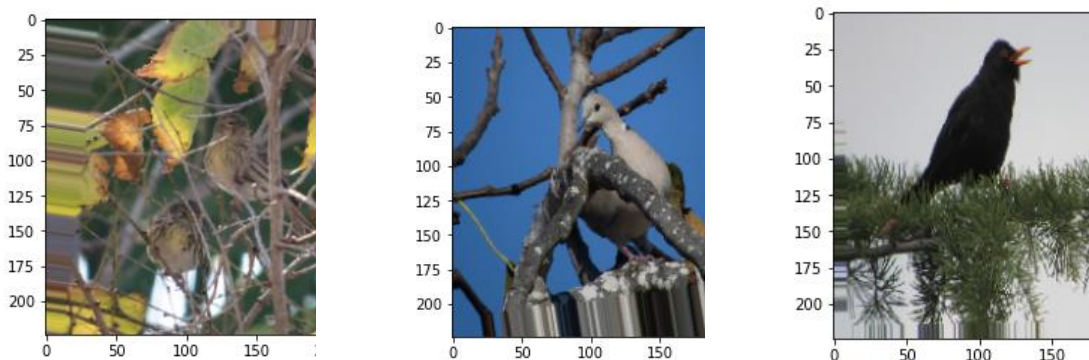


Ilustración 33:ejemplos transformaciones

4 Experimentación

Tras explicar la implementación se van a analizar y mostrar los resultados de las pruebas realizadas durante el desarrollo del proyecto. Como se ha explicado en el punto anterior primero se mostrará los resultados con Nabirds y posteriormente con el conjunto de datos que se ha realizado para este proyecto.

4.1 Nabirds

En este apartado se van a analizar los resultados obtenidos con la base de Nabirds con las 5 arquitecturas diferentes, dependiendo de sus resultados se verá que red se va a usar para solventar nuestra problemática.

Las pruebas se han realizado con las mismas condiciones para todas las arquitecturas, con el mismo procesado de imágenes, reescalando las imágenes a 224x224 y un *batch size* de 32. Para su entrenamiento se ha empleado el optimizar Adam por defecto, es decir 0.001, y 15 épocas.

En la *Tabla 3* podemos observar los resultados obtenidos, Analizándolos podemos observar como las dos redes que mejor entrenan han sido ResNet50 y la EficientNetB5. Pero al enfrentarse a la validación, Inception_Resnet_V2 y la EficientNetB5 son la que han dado un mejor resultado.

| | Train_loss | Train_accuracy | Val_loss | Val_accuracy |
|----------------------------|------------|----------------|----------|--------------|
| Vgg16 | 0.485 | 0.9171 | 1.407 | 0.830 |
| ResNet50 | 0.083 | 0.997 | 0.277 | 0.934 |
| Inception_Resnet_V2 | 0.1 | 0.967 | 0.1458 | 0.96 |
| MobileNet | 0.147 | 0.954 | 0.678 | 0.838 |
| EficientNetB5 | 0.016 | 0.998 | 0.059 | 0.986 |

Tabla 3: Resultados de entrenamiento y validación tras 15 épocas

A continuación, se mostrarán mediante gráficas una comparación entre el *Train_accuracy* -*Val_accuracy*, y *Train-Loss* -*Val-Loss*. Como se puede observar, en general, en todos los modelos el acierto va creciendo y el error reduciéndose de forma

exponencial. En el caso del *Val.* sí que vemos como es más variable, en algunos casos produciéndose casi *overfitting*.

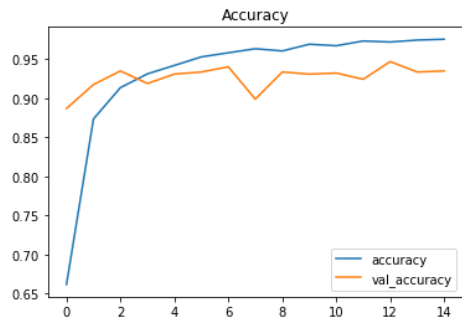


Ilustración 34: Entrenamiento ResNet50

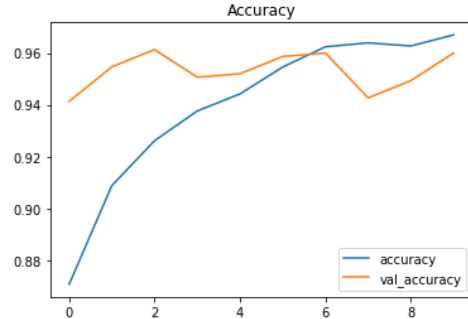


Ilustración 35: Entrenamiento Inception_Resnet_V2

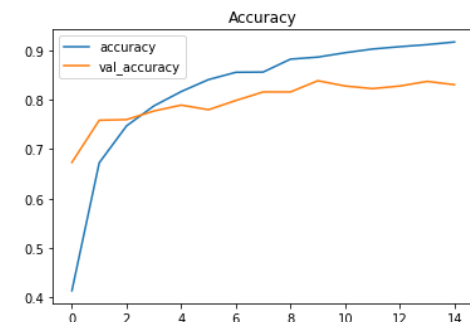
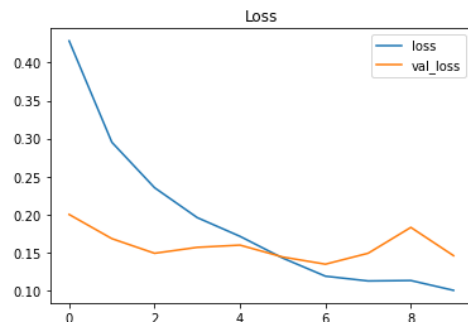
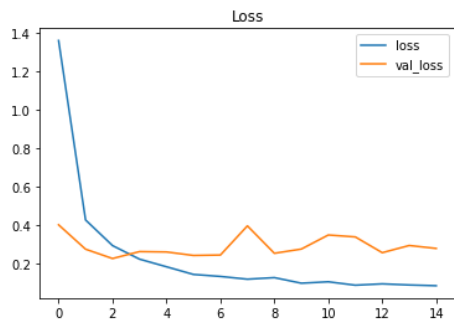


Ilustración 36: Entrenamiento VGG16

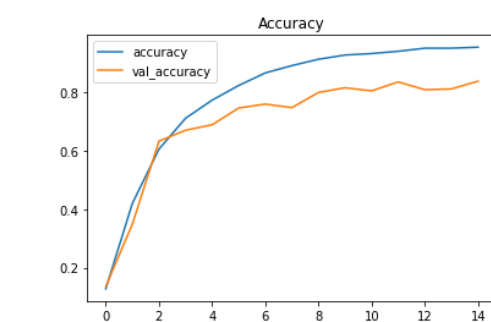
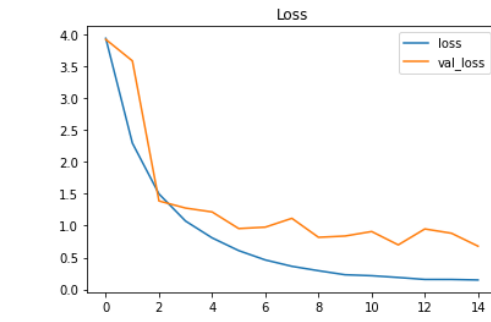
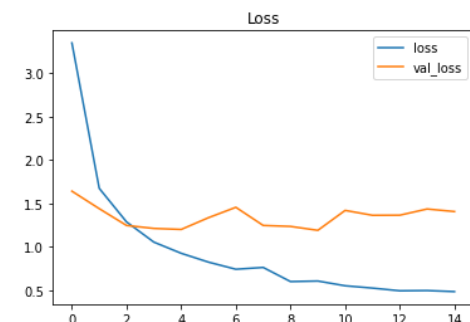


Ilustración 37: Entrenamiento MobileNet



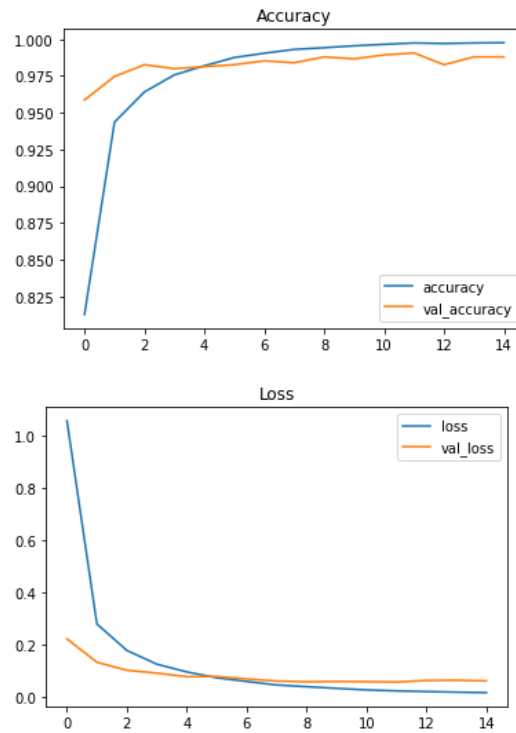


Ilustración 38: Entrenamiento *EfficientNetB5*

Tras ver los resultados obtenidos durante el entrenamiento, vamos a sacar su comportamiento ante el subconjunto de *test*. Las arquitecturas Resnet50 y EfficientNetB5 son las que mejores resultados han obtenido.

| test_accuracy | |
|----------------------------|-------|
| Vgg16 | 0.8 |
| ResNet50 | 0.958 |
| Inception_Resnet_V2 | 0.95 |
| MobileNet | 0.85 |
| EfficientNetB5 | 0.992 |

Tabla 4: Resultados Test_accuracy

A la hora de seleccionar un modelo se va a tener en cuenta los resultados val y test *accuracy*, viendo los porcentajes la siguiente gráfica podemos ver como los dos mejores modelos y los que vamos a usar para validar nuestro problema son: Inception_Resnet_V2 y EficientNetB5.

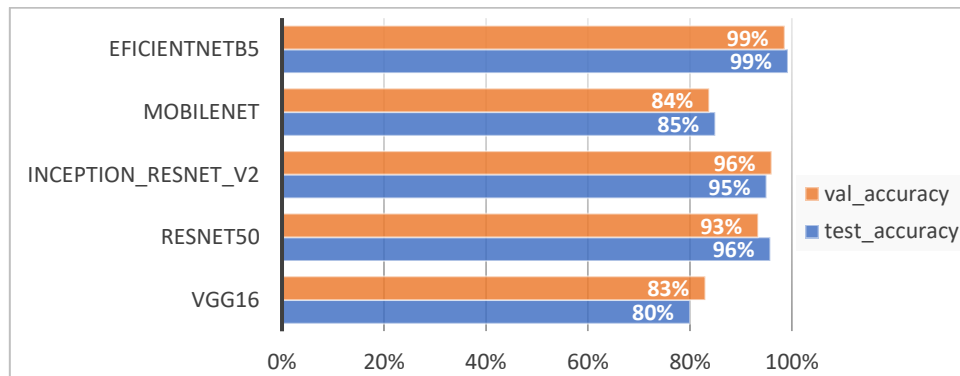


Ilustración 39: Porcentaje de Val Accuracy y Test accuracy

4.2 Base de datos propia

Tras haber hecho las pruebas necesarias con la base de datos de Nabirds, vamos a pasar a trabajar con el conjunto de datos realizado para este proyecto, en la *Ilustración 40* se pueden ver algunos ejemplos de las especies a clasificar.

Enlace: [montesinos8/TFG_alberto_montesinos: Repositorio de tfg "Reconocimiento visual de aves con Deep Learning" \(github.com\)](https://github.com/montesinos8/TFG_alberto_montesinos)



Ilustración 40: Ejemplos base de datos

Para estas pruebas se va a implementar el *Early stopping* con un *patience* de 10 para asegurarse que el modelo entrena lo máximo posible. En cuanto al procesamiento de las imágenes se va a realizar lo mismo que en las pruebas anteriores.

A partir de esto se han entrenado los dos modelos con las mismas condiciones, la Inception_Resnet_V2 ha entrenado durante 43 épocas hasta que se ha detenido y EfficientNetB5 20 épocas. En un principio la EfficientNetB5 obtiene unos mejores resultados como podemos ver en la *Tabla 5*

| | Train_loss | Train_accuracy | Val_loss | Val_accuracy |
|----------------------------|------------|----------------|----------|--------------|
| Inception_Resnet_V2 | 0.1680 | 0.9468 | 0.7256 | 0.8333 |
| EfficientNetB5 | 0.0949 | 0.9777 | 0.4665 | 0.8750 |

Tabla 5 : Resultados de entrenamiento y validación

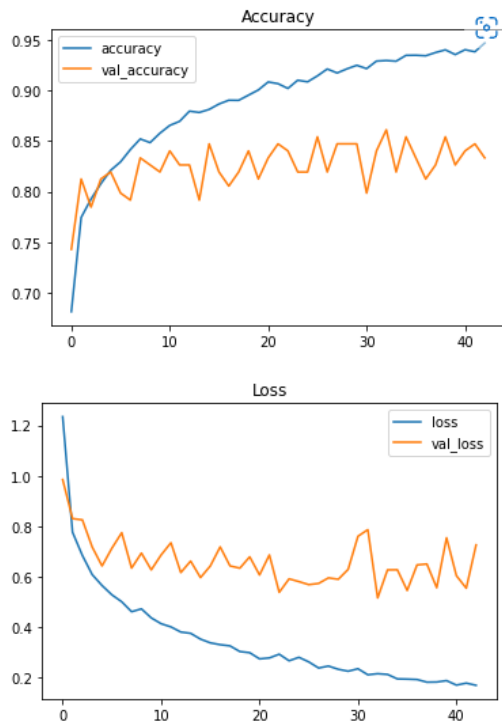


Ilustración 41: Entrenamiento Inception_Resnet_V2

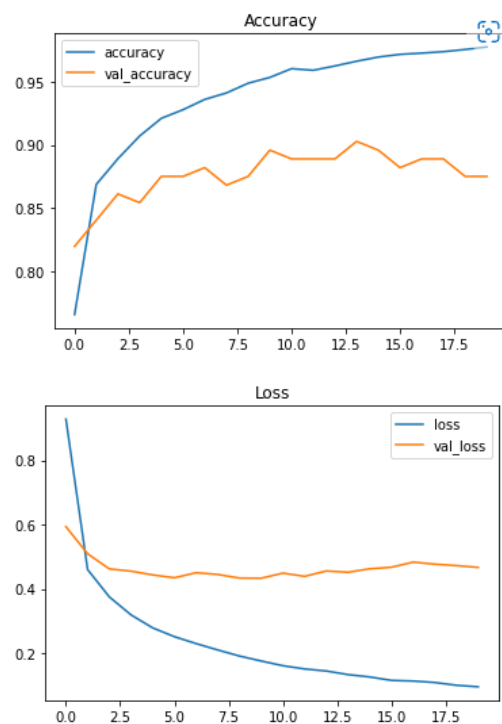


Ilustración 42: Entrenamiento EfficientNetB5

Una vez visto como han entrenado los dos modelos vamos a verificarlos con el conjunto de test. Como se puede observar la Inception_Resnet_V2 es la arquitectura que mejores resultados a obtenido.

| test_accuracy | |
|----------------------------|-------|
| Inception_Resnet_V2 | 0.95 |
| EfficientNetB5 | 0.948 |

Tabla 6: Resultados Tes_accuracy

4.3 Pruebas

En este apartado vamos a realizar pruebas con las imágenes que se obtuvieron con la cámara de fototrampeo que se dispuso en área del Parque de La Mata-Torre vieja.

El objetivo es ver si el modelo es capaz de clasificar correctamente estas imágenes, para ello probaremos con imágenes diferentes.

En esta primera prueba se ha buscado una imagen sencilla en la que solo se viera un ave totalmente centrada, en concreto una tórtola turca de la cual tenemos bastantes imágenes en entrenamiento. Como se puede observar en la *Ilustración 43* el modelo la ha etiquetado correctamente.

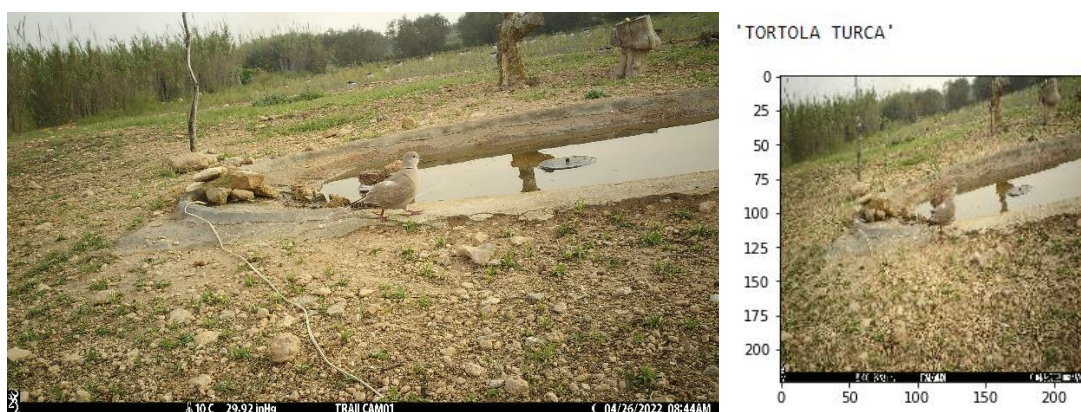


Ilustración 43: Prueba Tortola turca

El segundo caso es parecido al anterior, pero en este caso tenemos una Garcilla bueyera más alejada de la cámara. El modelo también funciona correctamente como se puede ver en la *Ilustración 44*.



Ilustración 44: Prueba Garcilla bueyera

Para las siguientes pruebas probaremos un ave negra que puede causar confusión entre varias especies y otra de tamaño pequeño. A continuación, se puede ver como en los dos casos el modelo vuelve a etiquetar correctamente.



Ilustración 45: Prueba Mirlo Común



Ilustración 46: Prueba verdecillo

Tras haber visto que el modelo funciona bien cuando aparece un ave en la imagen, vamos a evaluar lo que sucedería si aparecieran dos Garcillas bueyera por pantalla. Como se puede observar en la *Ilustración 47* lo clasifica bien.



Ilustración 47: Prueba misma especie

El siguiente caso de estudio que nos preocupa es que detecta el algoritmo cuando aparece más de una especie en la misma imagen, en la *Ilustración 48* podemos ver como etiqueta el ave más próxima a la cámara.



Ilustración 48: Prueba varias especies

5 Conclusión

Como conclusión del TFG se va a realizar un breve repaso a lo que se ha conseguido y aprendido, también se incluirá alguna breve reflexión sobre que mejoras se podrían implementar en un futuro.

Por un parte se ha conseguido tanto tratar y procesar bases de datos ya existentes, manejándolos y estandarizándolos de forma eficiente, como crear desde cero un conjunto de datos únicamente para este proyecto.

Debido en parte al desbalanceo de los datos se han conseguido implementar técnicas de *Data Augmentation* lo que ha mejorado rendimiento de los modelos estudiados.

También se han probado multitud de parámetros y arquitecturas ya existentes, con la finalidad de obtener la que mejor se ajustaba a nuestro objetivo. De estas pruebas han salido muy buenos resultados, mejor de los que se esperaban en algún momento de este proyecto.

A la hora de realizar pruebas con imágenes reales de la zona de estudio, los resultados ante imágenes que contiene un solo ave han sido muy positivos.

A pesar de que se han conseguido buenos resultados quedan aspectos en los que se podrían realizar mejoras. En cuanto al apartado de datos, se podría ampliar conjunto de dato, tanto el número de especies como variedad en las que ya se tienen, por ejemplo, más imágenes de aves volando. Una mejora que ayudaría mucho a la hora de entrenar sería etiquetar con *Bounding boxes* las imágenes del *dataset*, esto nos permitiría tener un mejor modelo.

El modelo también tiene mejoras que se pueden incorporar que no se han realizado por falta de tiempo o de conocimientos. Ahora mismo el modelo solo puede detectar un ave por imagen, sería necesario que se pudiera reconocer y clasificar todas las aves que se ven por pantalla.

En conclusión, considero que los resultados obtenidos han sido satisfactorios, ya que el modelo tiene buen rendimiento trabajando con imágenes con sólo ave y en parte se resuelve el problema planteado, empleando la inteligencia artificial para clasificar las aves de Áreas Naturales Protegidas

6 Referencias

- [1] Christian Szegedy, S. I. (2016). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*.
- [2] HUANG, Y.-P., & BASANTA, H. (n.d.). *Recognition of Endemic Bird Species Using Deep Learning Models*.
- [3] Norouzzadeha, M. S., A. N., Kosmala, M., & Swanson, A. (n.d.). *Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning*.
- [4] Schmidt, A., & van Swaay, C. (n.d.). *Improving the availability of data and information on species, habitats and sites*.
- [5] Villa, A. G., Salazar, A., & Vargas, F. (n.d.). *Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks*.
- [6] Willi, M., Pitman, R. T., Cardoso, A. W., Locke, C., & Swanson, A. (n.d.). *Identifying animal species in camera trap images using deep learning and citizen science*
- [7] [En línea] *The Cornell Lab*. (n.d.). Retrieved from <https://www.birds.cornell.edu/home/>
- [8] [En línea] *eBird*. (n.d.). Retrieved from <https://ebird.org/spain/home>
- [9] [En línea] *MerlinID*. (n.d.). Retrieved from <https://merlin.allaboutbirds.org/>
- [10] [En línea] *iNaturalist*. (n.d.). Retrieved from <https://www.inaturalist.org/home>
- [11] [En línea] *Netapp*. (n.d.). Retrieved from <https://www.netapp.com/es/artificial-intelligence/what-is-artificial-intelligence/>
- [12] Torres, D. J. (2020). Python Deep Learning: Introducción práctica con Keras y TensorFlow 2P.
- [13] Torres, D. J. (2019). Deep learning DEEP LEARNING Introducción práctica con Keras.

- [14] [En línea] *diegocalvo*. (n.d.). Retrieved from <https://www.diegocalvo.es/red-neuronal-convolucional/>
- [15] [En línea] G, R. (n.d.). *medium*. Retrieved from <https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918#:~:text=with%20transfer%20learning,-,VGG16%20Architecture,layers%20i.e.%2C%20learnable%20parameters%20layer.>
- [16] [En línea] *opengenus*. (n.d.). Retrieved from <https://iq.opengenus.org/resnet50-architecture/>
- [17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi (2016) *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*
- [18] [En línea] [En línea] *opengenus*. (n.d.). Retrieved from <https://iq.opengenus.org/mobilenet-v1-architecture>
- [19] Mingxing Tan, Q. V. (2020). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*.
- [20] [En línea] *Numpy*. (n.d.). Retrieved from <https://numpy.org/>
- [21] [En línea] *Pandas*. (n.d.). Retrieved from <https://pandas.pydata.org/>
- [22] [En línea] *Matplotlib*. (n.d.). Retrieved from <https://matplotlib.org/>
- [23] [En línea] *Scikr-learn*. (n.d.). Retrieved from <https://scikit-learn.org/stable/>
- [24] [En línea] *Tensorflow*. (n.d.). Retrieved from <https://www.tensorflow.org/>
- [25] [En línea] *Keras*. (n.d.). Retrieved from <https://keras.io/>
- [26] [En línea] *Jupyter*. (n.d.). Retrieved from <https://jupyter.org/>