Pontificia Universidad Católica del Ecuador

Facultad De Ingeniería

Sistemas De Información

TEMA:

Desarrollo de un modelo Deep Learning para reconocimiento de especies animales emblemáticas del Ecuador en peligro de extinción

AUTOR:

PAUL ALEJANDRO VALLEJO CABEZAS

QUITO, JULIO DE 2024

DEDICATORIA

El desarrollo de este proyecto va dedicado a mi familia, especialmente a mis padres, que su incondicional apoyo y soporte me ha otorgado la oportunidad de adquirir los conocimientos necesarios y, eventualmente, el reconocimiento profesional en esta institución. Su ininterrumpido interés por mi progreso ha sido crucial y un impulso en mi motivación para tratar de conseguir la excelencia en el aspecto académico y en mi desarrollo personal. Aunque se trata de un logro personal, el esfuerzo y papel que ha desempeñado cada miembro de mi familia y persona que me ha asistido durante el camino ha resultado en una contribución fundamental, por lo que este logro también les pertenece.

AGRADECIMIENTO

Mis padres han sido el pilar fundamental, emocional y económico que me ha facilitado la posibilidad de trabajar por mis metas, he logrado el desarrollo físico, emocional y mental gracias al soporte que me demuestran cada día, su incesable voluntad y dedicación solo pueden ser devueltos a través de mi profunda gratitud y el éxito que la vida me concederá el honor de enseñarles. Por lo pronto, cada esfuerzo y logro que alcance, es exclusivamente gracias a estos seres humanos.

RESUMEN

El desarrollo de un modelo de Deep Learning que permita la detección de especies animales específicas emblemáticas del Ecuador que se encuentran en peligro de extinción y sea capaz de clasificarlas. El modelo se construirá basado en la técnica de redes neuronales convolucionales (CNN), que se especializan concretamente en el reconocimiento de imágenes y video, dada su estructura inspirada en el funcionamiento de la corteza visual del cerebro humano. Las especies que serán abordadas son: el oso de antojos, el cóndor andino, el pingüino de Galápagos, el águila harpía y el jaguar. Se busca seguir el marco de trabajo de data mining CRISP-DM, que proporciona seis fases fundamentales para conseguir una minería de datos productiva y exitosa, sin embargo, debido a limitaciones de recursos se define el alcance a las cinco primeras fases, con lo que el proyecto concluirá en la evaluación del modelo. El presente tiene como fin aportar a la estrategia de conservación de la biodiversidad en Ecuador como herramienta de detección de fauna no invasiva.

ABSTRACT

The development of a Deep Learning model that allows the detection of specific animal species emblematic of Ecuador that are in danger of extinction and is able to classify them. The model will be built based on the technique of convolutional neural networks (CNN), which specialize specifically in image and video recognition, given its structure inspired by the functioning of the visual cortex of the human brain. The species to be addressed are: The Andean bear, the Andean condor, the Galapagos penguin, the harpy eagle and the jaguar. The aim is to follow the CRISP-DM data mining framework, which provides six fundamental phases to achieve productive and successful data mining; however, due to resource limitations, the scope is defined to the first five phases, and the project will conclude with the evaluation of the model. The purpose of this project is to contribute to the biodiversity conservation strategy in Ecuador as a non-invasive animal detection tool.

ÍNDICE

Contenido

ÍNDICE DE FIGURAS, GRÁFICOS Y TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1. Marco de referencia

1.1. Justificación

El reconocimiento de especies animales en vida silvestre es una actividad que se ha utilizado con fin de monitorear las poblaciones, identificar migraciones, invasiones y generalmente ejemplares mamíferos, con lo que se han podido desarrollar medidas eficientes de conservación de la biodiversidad.

Técnicas como el Deep Learning contribuyen significativamente en el reconocimiento de imágenes, surge la oportunidad de hacer uso de esta para identificar especies animales y evitar limitaciones de las metodologías actuales, aumentar la precisión y disminuir la invasión por nuestra parte. Por ello, con fin de desarrollar y evaluar el rendimiento del modelo construido, se busca seguir el marco de trabajo CRISP-DM que proporciona la estructura para llevar a cabo este proyecto, aunque se desarrollarán las cinco primeras fases únicamente, dejando el despliegue fuera, pero se asegurará una construcción y evaluación efectivos del modelo.

1.2. Planteamiento del problema

El Ecuador es considerado el país mega diverso más compacto del mundo, cuenta con una extensa cantidad de especies y muchas de ellas son endémicas del territorio, sin embargo, debido a destrucción de hábitats, la caza poco regulada, tráfico ilegal de especies o introducción de nuevas, varias de las especies animales emblemáticas, principalmente mamíferos y aves, se encuentran en peligro de extinción, por lo que es un área que requiere acción de carácter urgente.

Los métodos tradicionales de identificación de especies en vida silvestre comprenden una serie de limitaciones que han generado que esta actividad sea ineficiente, se presentan retos como las condiciones de observación, recursos de tiempo y dinero considerables, además de las

limitaciones propias del ser humano. Es por esto, que este proyecto pretende lograr un reconocimiento eficiente de determinadas especies animales a través de imágenes mediante el modelo de Deep Learning, y de esta manera proporcionar una herramienta de identificación no invasiva que aporte a la estrategia de conservación de la fauna.

Del planteamiento surge la pregunta principal de investigación:

- ¿Cómo optimizar la extracción y procesamiento del conjunto de datos para lograr un modelo de clasificación de especies preciso siguiendo el marco CRISP-DM?

Y las interrogantes secundarias:

- ¿Cuáles son las métricas de evaluación más significativas para determinar el rendimiento del modelo?
- ¿Cuál es el impacto del tamaño de conjunto de datos de entrenamiento respecto a la precisión del modelo?
- ¿Qué especies animales específicas serán abordadas en el estudio para la recopilación de datos correspondientes?

1.3. Objetivo General

Desarrollar un modelo de Deep Learning para el reconocimiento de especies animales en vida silvestre en peligro de extinción en Ecuador a través de imágenes, siguiendo las fases propuestas por el marco de CRISP-DM, con fin de aportar una herramienta de detección de fauna no invasiva.

1.4. Objetivos Específicos

Extraer la colección de datos que comprenderá imágenes de las especies animales correspondientes que servirán de entrenamiento del modelo.

Realizar un preprocesamiento meticuloso del conjunto de datos para priorizar la calidad de los mismos.

Construir el modelo utilizando técnicas de Deep Learning ajustando los parámetros según sea necesario.

Evaluar el modelo con datos del conjunto de prueba a través de métricas adecuadas para determinar la precisión del mismo.

1.5. Antecedentes

El fototrampeo es una técnica, cuyo propósito ha sido identificar comportamientos de especies animales, detectar poblaciones, y estudiar el estado de salud del ecosistema donde actúa tomando fotografías y videos de manera autónoma a través de sensores de calor y movimiento. Estos dispositivos tienen capacidad para almacenar una cantidad considerable de imágenes las mismas que son luego registradas, provocando que el esfuerzo por clasificarlas sea bastante formidable, es por esta razón que ha surgido la necesidad de desarrollar modelos basados en técnicas de Deep Learning para que puedan procesar esta información.

El construido sistema automático de censo y seguimiento de la biodiversidad usando técnicas de aprendizaje profundo, financiado por la Fundación Biodiversidad del Ministerio para la Transición Ecológica y el Reto Demográfico (MITECO), por ejemplo, funciona de modo que detecta la zona de interés a través de un mapa de calor, reconoce que existe una especie animal en la fotografía y finalmente la red neuronal lo clasifica y determina de qué especie se trata. (Adetunji, 2021)

En otro caso, SAS Innovate, en función de su deber por contribuir a un cambio positivo a través del desarrollo tecnológico, se ha puesto en colaboración con el Centro de Estudios de Galápagos (CGS) con fin de desarrollar un proyecto centrado en el reconocimiento facial de tortugas marítimas a través de técnicas de machine learning, cabe resaltar que su propósito no

se limita al reconocimiento de estas especies, sino también en el estado de salud y comportamientos. Al emplear estas herramientas se puede conseguir información de abundante valor que sirve esencialmente a la toma de decisiones informadas y por consiguiente a la estructuración de estrategias eficientes que sirvan de apoyo para la conservación de la biodiversidad en el Ecuador. (Bowen, 2023)

1.6. Alcance

El proyecto tendrá como enfoque el desarrollo y evaluación de un modelo basado en técnicas de Deep Learning para el reconocimiento de especies de aves y mamíferos en peligro de extinción a través de imágenes, se ha establecido como marco de trabajo CRISP-DM y se cubrirán las fases de entendimiento del negocio, entendimiento de los datos, preparación de los datos, modelado, evaluación y finalmente despliegue. Es importante mencionar que no se llevará a cabo la sexta fase que corresponde al despliegue en un entorno de producción ni integración con sistemas de información externos. Pues, por cuestión de recursos y tiempo, se ha limitado el presente a ser desarrollado y evaluado.

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2. Marco Teórico

2.1. Reconocimiento de especies con Deep Learning

El Deep Learning, una rama del aprendizaje automático ha revolucionado el procesamiento de imágenes en la computación. En el contexto de la conservación de la biodiversidad en Ecuador, el uso de técnicas de Deep Learning, como las redes neuronales convolucionales (CNN), es fundamental para el reconocimiento de especies animales en peligro de extinción. Las CNN son especialmente relevantes debido a su capacidad para emular el proceso de aprendizaje del cerebro humano y su habilidad para identificar formas y objetos dentro de las imágenes de manera automática y precisa.

La red neuronal convolucional funciona de modo que recibe como entrada tensores de forma, que comprende la estructura en la que están formados los datos, definiendo las dimensiones de las imágenes en pixeles y el número de canales de color, que vienen regidos en un dígito si la disposición de la imagen se encuentra en blanco y negro y en tres si contiene información de color RGB, en un rango de 0 a 255, esta regla de entrada se encuentra determinada en la capa de entrada que será la primera capa de convolución, es posible agregar tantas capas como la configuración del modelo requiera, dado que se declara como objeto de la clase Sequential que permite la aglomeración de capas.

Las CNN se distinguen de las redes neuronales convencionales por su habilidad en la detección de formas, texturas y patrones, no solamente de ubicación de los pixeles, para que esto ocurra esta red añade dos tipos de capas adicionales que son las de convolución y agrupación, que cumplirán con la extracción de características relevantes de las imágenes emulando el funcionamiento de nuestras neuronas simples y complejas en la corteza visual.

La convolución es una operación entre la imagen y un filtro, también llamado kernel o núcleo, en el que cada filtro aplicará transformaciones a través del cálculo de los pixeles y los valores autorregulados por este, con fin de obtener formas específicas. El número de filtros y el tamaño matricial de estos debe ser definido en cada capa, pues indicará con cuantos pixeles de la imagen se irá construyendo el mapa de características basado en la percepción de patrones básicos como bordes, orientaciones y texturas en posiciones, así como lo hacen las neuronas simples en nuestro cerebro al detectar características sencillas a través de campos receptores reducidos.

Las capas de agrupación, por otro lado, tienen campos receptivos más amplios que permiten la identificación de características generalizadas y más abstractas, como lo hacen las neuronas complejas de nuestra corteza. Esto se consigue por medio de la reducción de las dimensiones de las imágenes, recorriendo en zancadas del tamaño de la ventana de agrupación por los mapas generados en la convolución, esto se hace con el fin de registrar el valor más alto de la ventana y obtener una nueva imagen que preserve la información relevante y características importantes.

Los datos resultantes de la extracción de características por las capas de convolución y agrupamiento deben ser transformados a un vector de una dimensión para establecer una conexión entre las capas convolucionales y las densas completamente conectadas. Estas últimas se encuentran estructuradas de modo que cada neurona está conectada a cada neurona de la capa anterior, de esta manera, basado en la combinación de características extraídas son las responsables de disponer a la capa de salida una red procesada para poder tomar una decisión en la clasificación.

Tras el procesamiento se obtiene como salida la clase o la probabilidad de pertenecer a una de ellas, a través de la función de activación softmax. Con lo que, especies emblemáticas del Ecuador como el oso de anteojos, el cóndor andino, el águila harpía, el jaguar y el pingüino de Galápagos podrán ser procesadas e identificadas.

Figura 1.

Funcionamiento redes neuronales convolucionales

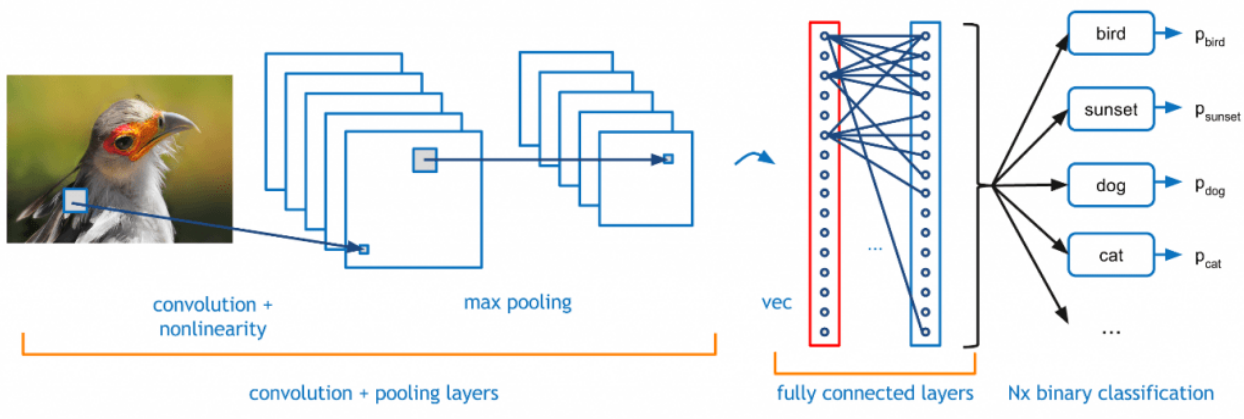


Figura No. 1 Redes Neuronales Convolucionales CNN (lisdatasolutions, s.f.)

2.2. Marco de trabajo CRISP-DM

CRISP-DM, que son las siglas de Cross-Industry Standard Process for Data Mining, es un método probado para orientar sus trabajos de minería de datos. Como metodología, incluye descripciones de las fases normales de un proyecto, las tareas necesarias en cada fase y una explicación de las relaciones entre las tareas. Como modelo de proceso, CRISP-DM ofrece un resumen del ciclo vital de minería de datos. (IBM, ibm.com, 2021)

El modelo de proceso CRISP-DM proporciona un ciclo de seis fases fundamentales para desarrollar un proyecto de minería de datos de forma sistemática, estas fases son: Comprensión del negocio, Comprensión de los datos, Preparación de los datos, Modelado, Evaluación y Despliegue. Aunque las fases se encuentran dispuestas en un ciclo, es posible avanzar y retroceder en las etapas según sea necesario.

Figura 2

Ciclo de vida de minería de datos

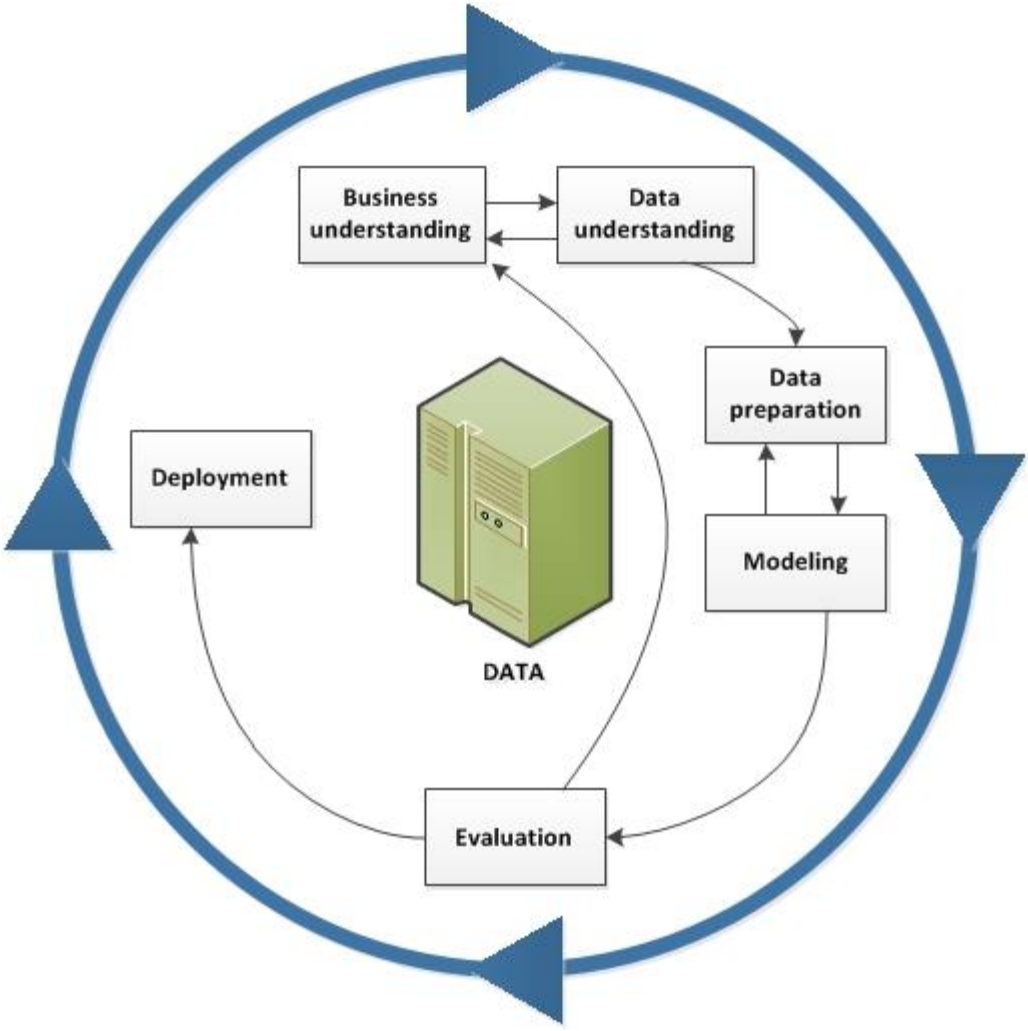


Figura No. 2 Fases Minería de Datos CRISP-DM (IBM, 2021)

2.3. Tecnologías y herramientas utilizadas

2.3.1. Google Colab

Colab es un servicio alojado de Jupyter Notebook que no requiere configuración y que ofrece acceso gratuito a recursos de computación, como GPUs y TPUs. Colab es una solución especialmente adecuada para el aprendizaje automático, la ciencia de datos y la educación. (Google, 2024)

La importancia del uso de este entorno radica en el aprovechamiento de recursos esenciales de pago, particularmente considerando la GPU A100 que es la mejor adaptada para llevar la construcción del modelo de manera eficiente.

2.3.2. Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales proporcionan un enfoque más escalable para las tareas de clasificación de imágenes y reconocimiento de objetos al aprovechar los principios del álgebra lineal, en concreto la multiplicación de matrices, para identificar patrones en una imagen. (IBM, ibm.com, 2023)

2.3.3. TensorFlow

TensorFlow es una librería de código libre para Machine Learning (ML). Fue desarrollado por Google para satisfacer las necesidades a partir de redes neuronales artificiales. TensorFlow te permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos. (Alonso, 2022)

2.3.4. Modelo de proceso CRISP-DM

Crisp-DM (Cross Industry-Standard Process for Data Mining) es una guía que proporciona un marco sólido para el desarrollo del ciclo de vida de la minería de datos, comprende seis fases que son: Comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación, y despliegue.

2.3.5. Lenguaje de programación Python

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. (aws, 2023)

2.3.6. Biblioteca Keras

Keras es una biblioteca de código abierto (con licencia MIT) escrita en Python. El objetivo de la biblioteca es acelerar la creación de redes neuronales: para ello, Keras no funciona como un framework independiente, sino como una interfaz de uso intuitivo (API) que permite acceder a varios frameworks de aprendizaje automático y desarrollarlos. (Cloud, 2024)

2.4. Repositorios de conjuntos de datos de especies animales

2.4.1. API de Custom Search JSON

La API de Custom Search JSON de Google te permite desarrollar sitios web y aplicaciones para recuperar y mostrar resultados de la búsqueda de Programmable Search Engine de manera programática. Con esta API, puedes usar las solicitudes RESTful para obtener resultados de búsqueda web o de búsqueda de imágenes en formato JSON. (Developers, 2023)

Google Cloud Platform ofrece una solución en modo de interfaz para la que se requiere adquirir una clave que permite acceder a la consola de desarrolladores de Google y la configuración de motores de búsqueda personalizados, a través de los cuales se realiza las solicitudes HTTP junto con las credenciales obtenidas y recibir los resultados en formato JSON, archivos de describen metadatos de cada imagen, incluyendo la URL. Una vez se obtengan los URLs de interés es necesario utilizar un script para manipular las JSON a través de solicitudes HTTP como Python y descargar las imágenes correspondientes.

2.4.2. Roboflow

Roboflow es una plataforma de inteligencia artificial avanzada especializada en modelos de videos e imágenes, que ofrece herramientas para la gestión de datos, anotación y entrenamiento de estos. Roboflow, que proporciona imágenes de alta calidad y modelos preentrenados, lo convierte en una fuente de datos esencial para este modelo.

CAPÍTULO III: DESARROLLO MODELO

3. Desarrollo y evaluación del modelo de Deep Learning

3.1. Definición de objetivos y requerimientos

3.1.1. Establecimiento de los objetivos específicos del modelo de RNC

Si bien la primera fase que propone CRISP-DM implica la identificación de objetivos comerciales que buscan ser cumplidos a través de la creación del modelo de minería de datos, al tratarse de un proyecto con fines académicos que no ha propuesto una empresa real o ficticia como objeto, se imposibilita la evaluación de la situación comercial en la que se considera recursos, criterios de rendimiento y riesgos para definir los objetivos en el ámbito. Por esta razón no serán abordados, pero sí objetivos de minería de datos.

Como bien se ha establecido, se busca desarrollar una herramienta de reconocimiento de imágenes de cinco especies animales en peligro de extinción emblemáticas del Ecuador, con lo que surgen los objetivos de minería de datos:

Construir un modelo de clasificación para el reconocimiento de especies animales en peligro de extinción emblemáticas de Ecuador a través de imágenes.

Conseguir una precisión mínima del modelo del 75% mediante las métricas de evaluación.

3.1.2. Identificación y análisis de los requerimientos del proyecto

Se ha definido como requerimiento funcional a la integración de librerías y APIs, y la evaluación de las herramientas que implica el desarrollo del modelo, por ende, se requiere del entorno de Jupyter Notebook proporcionado por Google Colab, dentro del que se consumirá la API de Google Custom Search para la extracción de los conjuntos de datos, construido para facilitar la recopilación de imágenes de especies animales. Además, una vez preparada la data adecuadamente, se llevará a cabo la construcción del modelo de redes neuronales

convolucionales utilizando la librería de tensorflow en conjunto con la interfaz de keras para la clasificación de imágenes y la evaluación del rendimiento a través de métricas.

Respecto a los requerimientos no funcionales, se toma en consideración una documentación del código exhaustiva para asegurar su comprensión y más importante, se precisa el licenciamiento de los servicios de Google, tanto para la generación de la API del motor de búsqueda Custom Search, como la herramienta Colab para garantizar un correcto, seguro y eficiente entrenamiento del modelo a través de la continuidad de ejecución y aprovechamiento de recursos exclusivos necesarios para el desarrollo, específicamente refiriéndose a la utilización de GPU A100 que es la más adecuada para el desarrollo.

3.1.3. Definición de los criterios de éxito y métricas de evaluación del modelo

Considerando que se trata de un desarrollo de un modelo que cuenta con escasos recursos para ser entrenado, además de la complejidad respecto al manejo meticuloso de los conjuntos de datos refiriéndose a la cantidad y calidad de fotografías recolectadas, los criterios de éxito deberán regirse al alcance del proyecto.

Siendo estos:

Umbral de precisión que el modelo debe alcanzar para ser considerado aceptable es 75%.

El contraste visual entre la imagen cargada al modelo y el resultado de clasificación para verificar exactitud del modelo o identificar oportunidades de mejora.

La eficiencia del modelo respecto al tiempo en que demora en procesar la imagen y clasificarla, donde se espera el menor posible.

Definiendo las métricas de evaluación a ser utilizadas:

Precisión o accuracy: Porción de la muestra que el modelo ha clasificado correctamente.

Sensibilidad o recall: Porción de casos positivos correctamente indentificados.

F1-score: Medida ponderada entre la precisión y el recall.

Matriz de confusión: Muestra resultados respecto a verdaderos positivos y negativos, y falsos positivos y negativos.

3.2. Comprensión de los Datos

3.2.1. Recopilación de los datos de los repositorios establecidos

Para la recolección de datos se manejó el motor de búsqueda personalizado, que se creó a partir de la API Google Custom Search. Se llevaron a cabo las consultas necesarias para cada una de las especies animales en peligro de extinción que se abordarán en el modelo consumiendo la API. Además, se utilizó la plataforma Roboflow para la adquisición de conjuntos de datos de especies animales para asegurar un equilibrio de cantidad de imágenes entre las clases. El procedimiento abarcó:

La creación de funciones para realizar las consultas, estableciendo parámetros de términos y número de solicitudes, y para la obtención de listas a partir de los links de respuesta en formato JSON, para cada una de las especies animales.

Implementación de bucles y modificación de parámetros de búsqueda para manejar las limitaciones de respuesta establecidos por la API.

Descarga de imágenes a partir de las listas de links de respuesta obtenidas y organizarlas en carpetas correspondientes a cada especie para almacenarlas en Google Drive. Las imágenes obtenidas de Roboflow serían guardadas en el mismo repositorio.

Figura 3

Función para consumir API de motor de búsqueda personalizado

```python
import requests, time

def buscar_imagenes(query, resultados_lista, consultas=20):
    # URL de Solicitud
    url = "https://www.googleapis.com/customsearch/v1"

    # Parámetros base
    params_base = {
        "q": query,  # Términos de búsqueda
        "cx": "9526749cc47b0453e",  # ID de búsqueda personalizada (CX)
        "key": "AIzaSyBsS_yi83vN61c1GSJUpgRAUd3UTi5WIlo",  # Clave de API
        "searchType": "image",  # Tipo de búsqueda: imágenes
        "num": 10  # Número de resultados (máximo)
    }

    # Utilizar un conjunto para URLs únicas
    urls_unicas = set()

    # Realizar consultas
    for i in range(consultas):
        # Ajustar el parámetro start
        params = params_base.copy()
        params["start"] = i * params["num"] + 1

        # Solicitud HTTP
        response = requests.get(url, params=params)

        # Si la solicitud fue exitosa
        if response.status_code == 200:
            # Guardar respuesta JSON
            json_data = response.json()
            # Procesar el JSON para obtener las URL de las imágenes
            items = json_data.get("items", [])
            # Agregar URLs únicas a la lista de resultados
            for item in items:
                link = item.get("link")
                if link not in resultados_lista:
                    resultados_lista.append(link)
```

Figura No. 3 Función Custom Search API

Los términos de búsqueda y las consultas fueron establecidas de modo que el retorno alcance alrededor de mil enlaces por cada especie. Aunque se ha ponderado la posibilidad de obtener respuestas de baja calidad, se espera conservar una gran fracción de datos relevantes.

Figura 4

Consultas para cada especie

```
def imprimir_lista(lista, queries):
  # 20 Consultas (200 links de imágenes) por cada query
  for query in queries:
    buscar_imagenes(query, lista)

  print("Imágenes en la lista:", len(lista))
  print(lista)
```

Establecer queries y realizar consultas de cada especie

```
# Lista de links de Pinguinos de Galápagos
pinguinos = []
# Distintos queries para controlar la variación de respuestas
queries = ["pingüino de Galápagos", "pinguino de Galápagos", "Spheniscus mendiculus", "Galápagos penguin",
           "mendiculus", "pingüino de Ecuador", "Galápagos Islands penguin", "pingüino Galápagos foto",
           "Galapagos penguin photos", "spheniscus mendiculus photos", "pinguino tropical ecuatorial"]

imprimir_lista(pinguinos, queries)
```

```
Imágenes en la lista: 1049
['https://upload.wikimedia.org/wikipedia/commons/2/26/Galapagos_Penguin_%2846860487615%29.jpg', 'https://cdn.dow
```

4 Consultas API para cada especie

3.2.2. Análisis exploratorio de imágenes

Los científicos de datos utilizan el análisis de datos exploratorios (EDA) para analizar e investigar conjuntos de datos y resumir sus características principales, a menudo empleando métodos de visualización de datos. (IBM, 2024)

Con fin de realizar un análisis exploratorio provechoso de los conjuntos de datos, es preciso definir los aspectos relevantes de estos para el estudio, como cantidad y calidad de las imágenes. Tras la familiarización con los datos a través de cálculos y gráficos de visualización como histogramas, se obtuvieron los resultados sobre sus características.

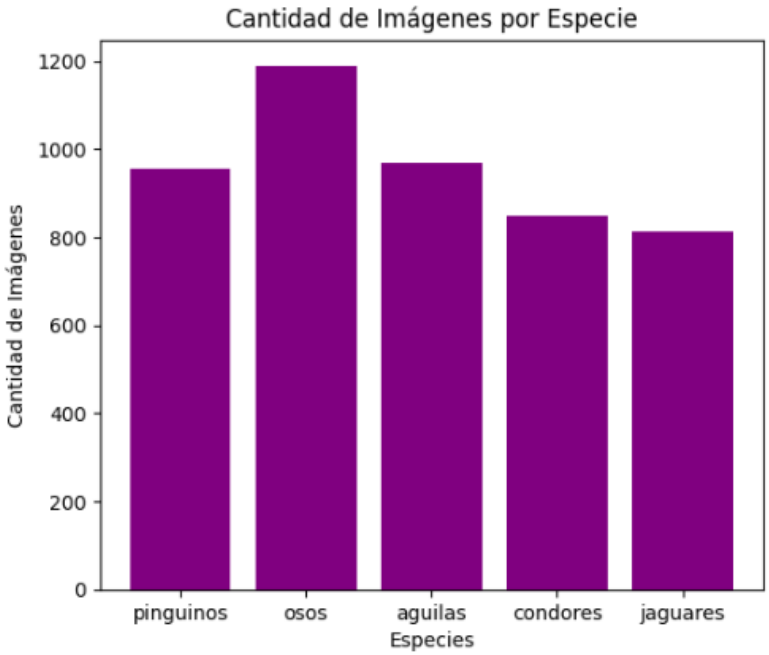
Tamaño del conjunto de datos. La suma de las imágenes por especie animal terminó en un tamaño total de 4776 imágenes, teniendo como cantidad de imágenes por clase:

- Pingüinos: 956 imágenes
- Osos: 1189 imágenes

- Águilas: 970 imágenes
- Cóndores: 849 imágenes
- Jaguares: 812 imágenes

- **Figura 5**

  Gráfico de barras de cantidad de imágenes por especie



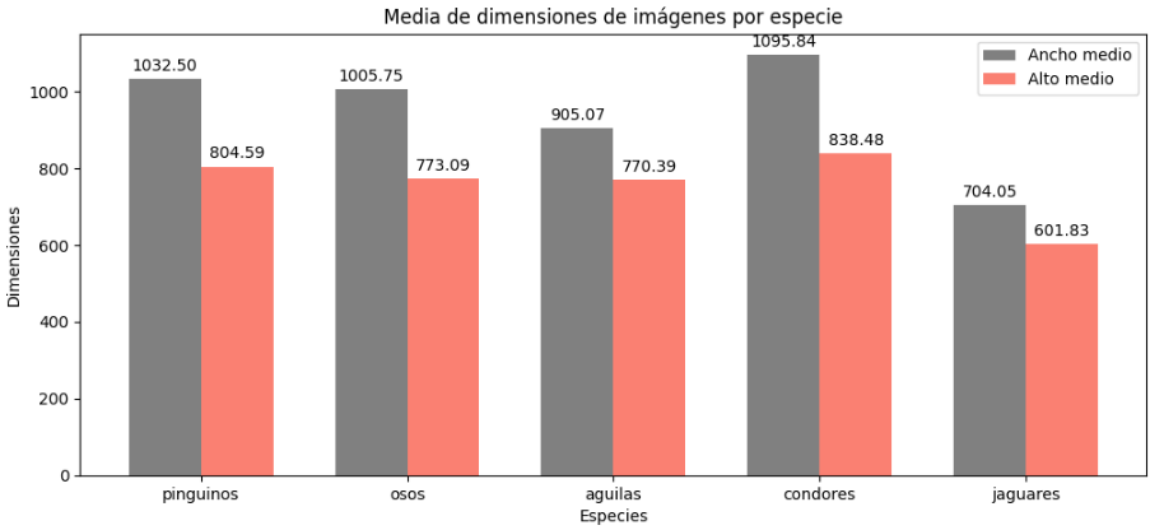
5 Cantidad de imágenes por especie

Tamaño de las imágenes. Conocer las dimensiones de las imágenes puede contribuir a determinar la transformación de tamaño de las imágenes antes de usarlas en la red. Promedio de las dimensiones en pixeles de ancho y alto:

- Media de dimensiones en pingüinos: 1008.54 x 837.71
- Media de dimensiones en osos: 947.37 x 800.01
- Media de dimensiones en águilas: 908.77 x 805.28
- Media de dimensiones en cóndores: 1050.18 x 857.55
- Media de dimensiones en jaguares: 988.64 x 864.72

Figura 6

Dimensiones promedio de imágenes



6 Promedio de dimensiones de datos

Pixeles de color y su variabilidad. Para llevar a cabo este análisis y manejar la complejidad del mismo, se ha tomado como muestra a las cinco primeras imágenes del conjunto de 'pingüinos', demostrando a través de histogramas, características que comparten y también presentando la variabilidad de pixeles en las imágenes, que ayudan a definir una normalización de pixeles de valores en un rango entre 1 a 0.

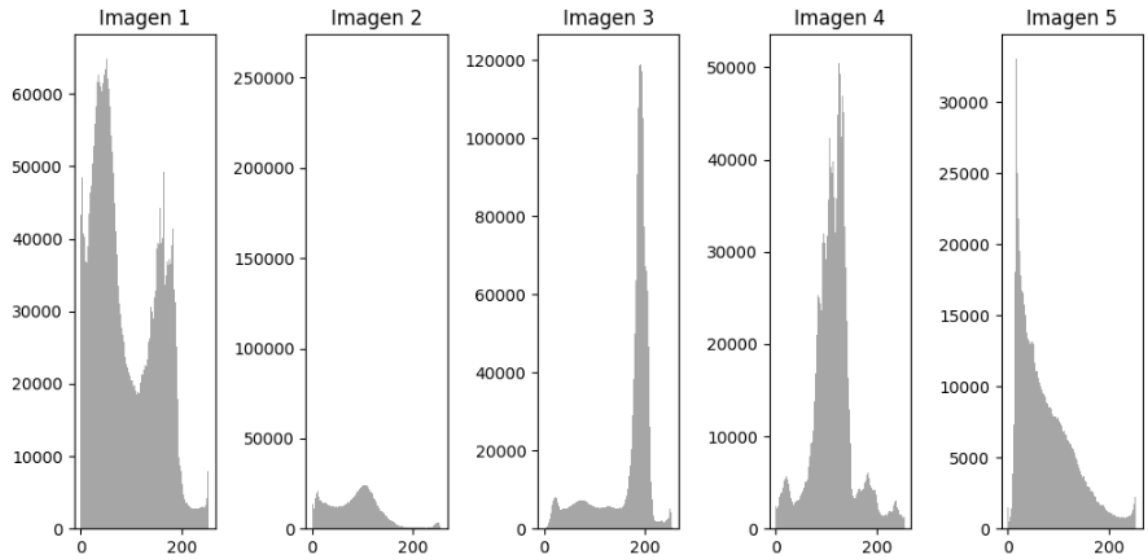
Figura 7

Primeras imágenes de la carpeta pingüinos

7 Muestra de imágenes de carpeta pingüinos

Figura 8

Variabilidad de las cinco primeras imágenes de ‘pinguinos’



8 Variabilidad de pixeles en las imágenes de la carpeta 'pinguinos'

3.2.3. Identificación de problemas o requerimientos de transformación

Tras la visualización de muestras de los conjuntos de datos se ha determinado la obligatoriedad de ciertos ajustes en la disposición de las imágenes. Primordialmente, se debe dar paso a la selección de imágenes de relevancia, eliminando del conjunto los datos que no aportan al desarrollo del modelo o incluso entorpecerían su desempeño. El conjunto de datos apto debe seguir las siguientes consideraciones:

Se busca que las imágenes tengan las mismas dimensiones, dado que las entradas de las redes neuronales así lo requieren. Estableciendo su tamaño a un estándar de 256 x 256, reduciendo la cantidad de datos que deben ser procesados evitando la pérdida de información.

Los valores de pixeles de cada imagen tendrán que ser normalizados, pudiendo ser presentados en rangos de 0 a 255, manteniendo la consistencia de datos y, por ende, la estabilidad del modelo.

Dada la cantidad considerablemente baja de cada conjunto de datos, se deben aplicar técnicas de aumentación de datos, tales como la rotación, con fin de evitar el sobreajuste y garantizar una mayor robustez.

Aplicar técnicas de eliminación de ruido como el filtro de la mediana para aumentar la calidad.

3.3. Preparación de los Datos

3.3.1. Selección de imágenes para su procesamiento

Dado el origen de las imágenes recolectadas, se presentó una gran probabilidad de la presencia de imágenes poco relevantes para el modelo, considerando que solo deben ser tomadas en cuenta fotografías reales de cada especie animal, se utilizó un método interactivo usando widgets con fin de visualizar y eliminar las imágenes no deseadas de manera manual.

Figura 9

Método a través de widgets para la eliminación de imágenes.

9 Eliminación de imágenes irrelevantes

Adicionalmente, con fin de asegurar un conjunto de datos de alta calidad, no pueden permitirse elementos repetidos dentro de los conjuntos. Existe una manera que garantiza una identificación de elementos idénticos a través de la generación de un resumen de 256 bits por imagen utilizando el algoritmo SHA-256, así si otra imagen contiene el mismo contenido hash, implicaría duplicidad y tendrá que ser eliminada la copia.

Figura 10

Función para calcular contenido hash de imágenes

```
def calcular_hash(archivo):
    hasher = hashlib.sha256()
    with open(archivo, 'rb') as f:
        buf = f.read()
        hasher.update(buf)
    return hasher.hexdigest()
```

10 HASH para identificar imágenes duplicadas

Tras la selección de datos de datos de valor se obtuvieron los resultados.

- Cantidad de imágenes en pingüinos: 532
- Cantidad de imágenes en osos: 506
- Cantidad de imágenes en águilas: 487
- Cantidad de imágenes en cóndores: 532
- Cantidad de imágenes en jaguares: 500

Las imágenes que serán procesadas, serán almacenadas en nuevos directorios para llevar a cabo las transformaciones correspondientes y mantener las originales libres de cambios.

3.3.2. Transformación y limpieza de datos

Siguiendo las consideraciones propuestas en la disposición de los datos, las dimensiones y canales de color de las imágenes deben ser alterados obligatoriamente. Reducir a dimensiones muy pequeñas aceleraría el procesamiento, sin embargo, es imprescindible evitar la pérdida de

información para que el modelo sea entrenado adecuadamente, por lo que se definirá un ancho de 256px y un alto de 256px para cada imagen, estándar mandatorio para la capa de entrada.

Figura 11

Función para redimensionar las imágenes de todas las carpetas

```python
# Función para redimensionar imágenes
def redimensionar_imagenes(carpeta):
    tamaño = (256, 256)
    for nombre_imagen in os.listdir(carpeta):
        ruta_imagen = os.path.join(carpeta, nombre_imagen)
        imagen = cv2.imread(ruta_imagen)
        if imagen is not None:
            imagen_redimensionada = cv2.resize(imagen, tamaño) # Redimensión de la imagen
            cv2.imwrite(ruta_imagen, imagen_redimensionada)
```

```python
# Aplicar redimensionamiento
for carpeta in carpetas:
    destino = os.path.join(ruta_procesada, carpeta)
    redimensionar_imagenes(destino)
    print('Imágenes redimensionadas en carpeta:', carpeta)
```

11 Función para redimensionamiento de imágenes

El redimensionamiento realizado da paso a la normalización de pixeles de color para estandarizar su procesamiento y que sus valores se presenten en un rango de 0 a 1, en lugar de 0 a 255. Para guardar esta configuración se requiere de una transformación de tipos de archivo actuales (.jpg) a un formato numpy (.npy) que sea capaz de almacenar esta información.

Además, en esta etapa se presenta también una bifurcación en el procesamiento de las imágenes. Se ha decidido que se precisa de modelos entrenados con conjuntos de datos con un solo canal de color (escala de grises), o tres canales (RGB), de esta manera distinguir el desempeño del modelo respecto a la diferencia de información.

De este modo, los archivos numpy que contendrán la información de las imágenes a color y escala de grises, sufrirán la modificación de conversión puntual respectivamente de ‘BGR’ a ‘Gray’, en el caso de escala de grises, o en su contrario a ‘RBG’, para asignar una disposición

correcta de color dado que el formato original indicaba un orden inadecuado de canales con el canal azul como primero.

Figura 12

Función para normalizar pixeles y transformar canal de color a RGB

```python
# Función para normalizar imágenes
def normalizar_imagenes(carpeta):
    for nombre_imagen in os.listdir(carpeta):
        ruta_imagen = os.path.join(carpeta, nombre_imagen)
        imagen = cv2.imread(ruta_imagen)
        if imagen is not None:
            imagen_rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB) # Transformar de BGR a RGB
            imagen_normalizada = imagen_rgb / 255.0 # Rango de 0 a 1
            np.save(ruta_imagen.replace('.jpg', '.npy'), imagen_normalizada)
```

12 Normalizar pixeles y transformar a canal de color RGB

Figura 13

Imagen redimensionada y normalizada en RGB

13 Imagen redimensionada y normalizada en tres canales de color

El conjunto de escala de grises fue modificado correspondientemente alterando la conversión de color, obteniendo la forma requerida para proceder.

3.3.3. Aplicación de técnicas de aumento de datos

La aumentación de imágenes o datos es una técnica usualmente utilizada para generar una mayor variabilidad de los datos, así también cuando no se dispone de una cantidad apropiada para llevar un entrenamiento productivo.

Esta técnica consiste en tomar los datos existentes en el dataset y crear variaciones a partir de modificaciones en la rotación, el desplazamiento horizontal o vertical, cortes y zoom aleatorios de la imagen, inversión horizontal y llenado de pixeles. Estos parámetros son regulables y pueden determinar la calidad del conjunto total si no son alterados convenientemente, ya que su mala configuración puede producir imágenes poco identificables.

Figura 14

Función para aumentar los conjuntos de datos .npy

```python
def aumentar_imagenes(carpeta, num=3):
    generador = ImageDataGenerator(
        rotation_range=180,  # Rotación aleatoria hasta ±180 grados
        width_shift_range=0.15,  # Desplazamiento horizontal hasta 15% del ancho
        height_shift_range=0.15,  # Desplazamiento vertical hasta 15% de la altura
        shear_range=0.2,  # Corte aleatorio hasta 20 grados
        zoom_range=0.25,  # Zoom aleatorio hasta 25%
        horizontal_flip=True,  # Inversión horizontal aleatoria
        fill_mode='nearest'  # Llenado de pixeles al más cercano
    )
    for nombre_imagen in os.listdir(carpeta):
        ruta_imagen = os.path.join(carpeta, nombre_imagen)
        imagen = np.load(ruta_imagen)
        # Expandir dimensiones para que sea compatible con el generador
        imagen = np.expand_dims(imagen, axis=0)
        contador = 0
        for imagen_aumentada in generador.flow(imagen, batch_size=1):
            imagen_aumentada = imagen_aumentada[0]  # Reducir dimensiones después de la aumentación
            nueva_imagen = f'aumentada_{contador}_{nombre_imagen}'
            ruta_nueva = os.path.join(carpeta, nueva_imagen)
            np.save(ruta_nueva, imagen_aumentada)
            contador += 1
            if contador >= num:
                    break
```

14 Aumentación de imágenes .npy

Para cada uno de los conjuntos se busca cuadruplicar el número de datos, por lo que se generarán tres variaciones por imagen. Además, es importante cerciorar que el conjunto de datos contenga las imágenes que serán aumentadas por lo que los archivos en formato .jpg serán eliminados.

Figura 12

Imágenes aumentadas de la carpeta cóndores

15 Muestra de imágenes aumentadas

Tras la aumentación el conjunto resultó en una cantidad de 10228 imágenes por las cinco especies animales, teniendo por cada una:

- Cantidad de imágenes en pingüinos: 2128
- Cantidad de imágenes en osos: 2020
- Cantidad de imágenes en águilas: 1948
- Cantidad de imágenes en cóndores: 2132
- Cantidad de imágenes en jaguares: 2000

El tamaño reflejado del conjunto de datos ya permite el entrenamiento de un modelo potencialmente preciso. Adicionalmente, se han evaluado técnicas de eliminación de ruido como el filtrado de media y el filtrado bilateral, sin embargo, debido a la complejidad de los datos y la cantidad de imágenes se ha decidido no llevar las transformaciones de suavizado ya que implica una pérdida de información que en esta ocasión no puede ser permitida.

3.3.4. Asignación de etiquetas

El modelo requiere que las imágenes se encuentren etiquetadas, es decir, el valor que se busca clasificar al procesar una entrada. Para ello se han cargado las imágenes en una lista X[] para iterarla, asignar la etiqueta correspondiente en y[], y convertirlas en arreglos numpy. Por ello las consideraciones de entrada son imperativas, que definen la forma de cada imagen en dimensiones de 256 y un canal de color, para escala de grises o tres canales si se trata de RGB.

Figura 16

Arreglos numpy de imágenes y etiquetas

```
print("Forma de X:", X.shape)
print("Forma de y:", y.shape)
```

```
Forma de X: (10228, 256, 256, 3)
Forma de y: (10228,)
```

```
print("Forma de X:", X.shape)
print("Forma de y:", y.shape)
```

```
Forma de X: (10228, 256, 256, 1)
Forma de y: (10228,)
```

16 Arreglos numpy con imágenes, forma y etiquetas

Al tratase de un modelo de clasificación que utiliza la función de pérdida de entropía cruzada categórica en la compilación del modelo, es mandatorio que las etiquetas se presenten en formato OneHotEncoding que es representado por un vector binario que define la clase.

Figura 17

Imágenes de cada clase y etiquetas one hot encoding

17 *Imágenes de cada clase con etiqueta*

3.4. Construcción del modelo CNN

3.4.1. Selección de técnicas de modelado

La técnica de modelado se ha determinado en base al objetivo de minería de datos del estudio, con fin de reconocer y clasificar imágenes de especies animales, se optó por la construcción de redes neuronales convolucionales que actuará de modelo de lenguaje supervisado, lo que implica que el conjunto de datos se encuentre etiquetado y sea dividido en conjuntos de entrenamiento y prueba para realizar las predicciones y con ello evaluar la precisión del modelo.

El conjunto de datos ha sido procesado con los requerimientos que el entrenamiento y evaluación demandan, la división en entrenamiento y prueba se ha establecido en un 85% (8694 imágenes) y 15% (1534 imágenes) correspondientemente tomando en consideración que durante el entrenamiento se tomará un 15% de los datos de entrenamiento como conjunto de validación que actúa de evaluador del rendimiento del modelo a medida que va siendo entrenado.

Se considerará la precisión en el conjunto de validación cuando el modelo termine de ser entrenado para considerar nuevas configuraciones en la arquitectura del modelo o modificaciones en los hiperparámetros hasta conseguir un resultado aceptable.

3.4.2. Generación del modelo

Para la construcción del modelo se han explorado varias configuraciones, arquitecturas y técnicas de regularización para evitar el sobreajuste y a través de la modificación esperar una precisión y generalización mejorada.

- Capas de Convolución: Usa filtros en la imagen de entrada para obtener características. (16, 32, 64), cada una con función de activación ReLU y campos receptivos de 3x3. La entrada con los parámetros de forma establecida (256, 256, 1 o 3).
- Capas de Max Pooling: Reducen el tamaño de la representación cada imagen a 2x2.

- Función de Activación ReLU: Usada por su eficiencia y control de mitigación del problema de gradiente usada en capas de convolución y capa densa.
- Dropout: Evita el sobreajuste apagando aleatoriamente la mitad de las neuronas durante el entrenamiento en valor 0.5.
- Regularización L2: Previene el sobreajuste penalizando los pesos grandes en 0.001. Aplicada en todas las capas del modelo.
- Batch Normalization: Genera una normalización de las entradas de las capas intermedias lo que estabiliza y hace más rápido el entrenamiento.
- Capa Flatten: Convierte los datos de las capas convolucionales a un vector unidimensional.
- Capa Densa completamente conectada: Toma el vector unidimensional y opera sus valores de manera lineal aplicando función no lineal ReLU en cada neurona.
- Función de Activación Softmax: Aplicada en la capa de salida para la distribución de probabilidad de las 5 clases.
- Early Stopping: Interrumpe el entrenamiento cuando el modelo en su conjunto de validación ya no ha mejorado con paciencia de 10 épocas.
- Optimizador Adam: Algoritmo en la complicación para optimizar los pesos durante entrenamiento.
- Pérdida categorical_crossentropy: Función de pérdida mide la diferencia entre la predicción y el valor real de la clase.
- Épocas y Batch Size: Iteraciones en el conjunto de datos (100 épocas) en lotes de 32.
- Callbacks: Definido por el early stopping y el almacenamiento de los logs tensorboard.

Técnicas como el Drop Out, L2 Regularization y el Batch Normalization fueron agregadas gradualmente para mejorar el rendimiento del modelo y evitar el sobreajuste. Además, la capa de entrada sufre una alteración para el conjunto de datos en escala de grises al definir un solo canal de color o tres canales para el caso de RGB. En total se estableció el entrenamiento de cinco modelos para el conjunto de escala de grises y dos para el conjunto de elementos a color.

Figura 18

Compilación y entrenamiento del modelo definido

```
[ ]  # Compilar el modelo
     modelo_batch.compile(optimizer='adam',
                          loss='categorical_crossentropy',
                          metrics=['accuracy'])
```

Entrenamiento del modelo

```
[ ]  modelo_batch.fit(X_train, y_train,
                      epochs=100,
                      batch_size=32,
                      validation_split=0.15,
                      callbacks=[tensorboard_batch])
```

18 Compilación y entrenamiento de modelo definido

Los modelos presentaron una precisión excelente en el conjunto de entrenamiento, pero una aceptable en el de validación determinando así una generalización de tres cuartas partes de los datos durante el entrenamiento. Los modelos con mejor rendimiento fueron guardados en Drive con formato HDF5 con fin de salvar el estado y poder utilizarlos a través de su importación.

Figura 19

Época 100 del mejor modelo conseguido RGB

```
Epoch 100/100
231/231 [==============================] - 5s 23ms/step - loss: 1.4130 - accuracy: 0.9555 - val_loss: 2.3424 - val_accuracy: 0.7531
<keras.src.callbacks.History at 0x7b9d24151d20>
```

19 Época 100 del mejor modelo entrenado

3.5. Evaluación del modelo

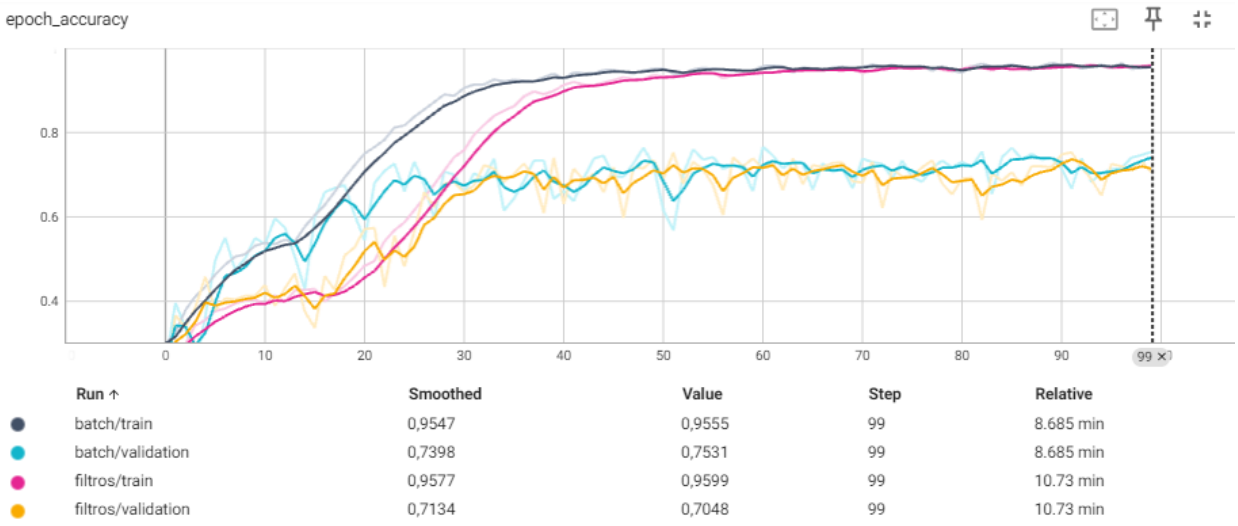
3.5.1. Visualización de rendimiento del modelo

Tensorboard es una herramienta de graficación que proporciona tensorflow para poder tener una visualización del rendimiento del modelo en cada una de las épocas respecto a la precisión y la pérdida en el conjunto de entrenamiento y validación. Considerando la precisión de validación podemos obtener una idea general del rendimiento global del modelo. Estos resultados son almacenados tras cada iteración en la definición del callback donde se parametrizó con fin de enviar los logs.

La comparación entre los modelos RGB se dispone de la manera:

Figura 20

Comparación de rendimiento entrenamiento y validación modelos RGB

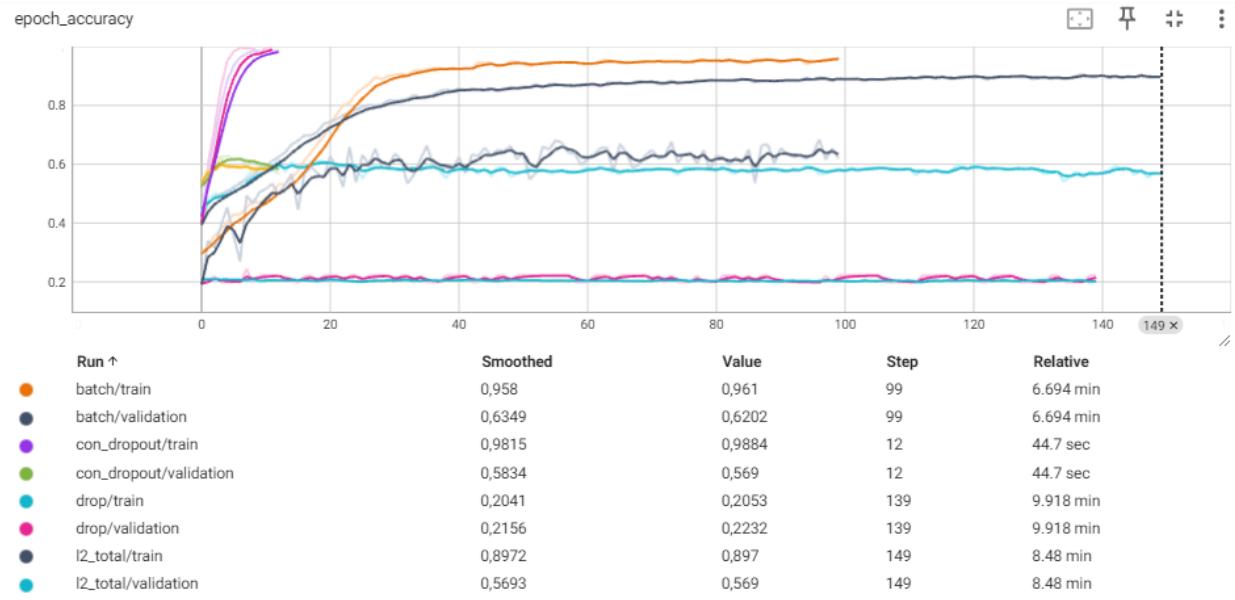


20 Comparación de rendimiento en entrenamiento y validación RGB

Los modelos en escala de grises consiguieron un rendimiento considerablemente inferior, el early stopping detuvo la mayoría de estos por la inexistente capacidad de mejora como se verá reflejado en el gráfico, mostrándose de la forma:

Figura 21

Comparación de rendimiento en entrenamiento y validación de modelo en grises



Run	Smoothed	Value	Step	Relative
batch/train	0,958	0,961	99	6.694 min
batch/validation	0,6349	0,6202	99	6.694 min
con_dropout/train	0,9815	0,9884	12	44.7 sec
con_dropout/validation	0,5834	0,569	12	44.7 sec
drop/train	0,2041	0,2053	139	9.918 min
drop/validation	0,2156	0,2232	139	9.918 min
l2_total/train	0,8972	0,897	149	8.48 min
l2_total/validation	0,5693	0,569	149	8.48 min

21 Comparación de rendimiento en entrenamiento y validación en grises

3.5.2. Visualización de predicciones

Con fin de observar el rendimiento real del mejor modelo se utilizó el conjunto de pruebas destinado para este fin, así poder realizar predicciones y obtener una visión general de la discrepancia entre la clase predicha y la real.

Con esto, al tener como salida etiquetas binarias, estas deben ser transformadas a clases discretas que se reflejarán como el valor de la etiqueta antes de convertirse a OneHotEncoding, esto permitirá el mapeo para imprimir cada clase como el nombre de la especie como tal y conseguir una mejor comprensión de las salidas.

Figura 22

Obtención de valores discretos a partir de las predicciones

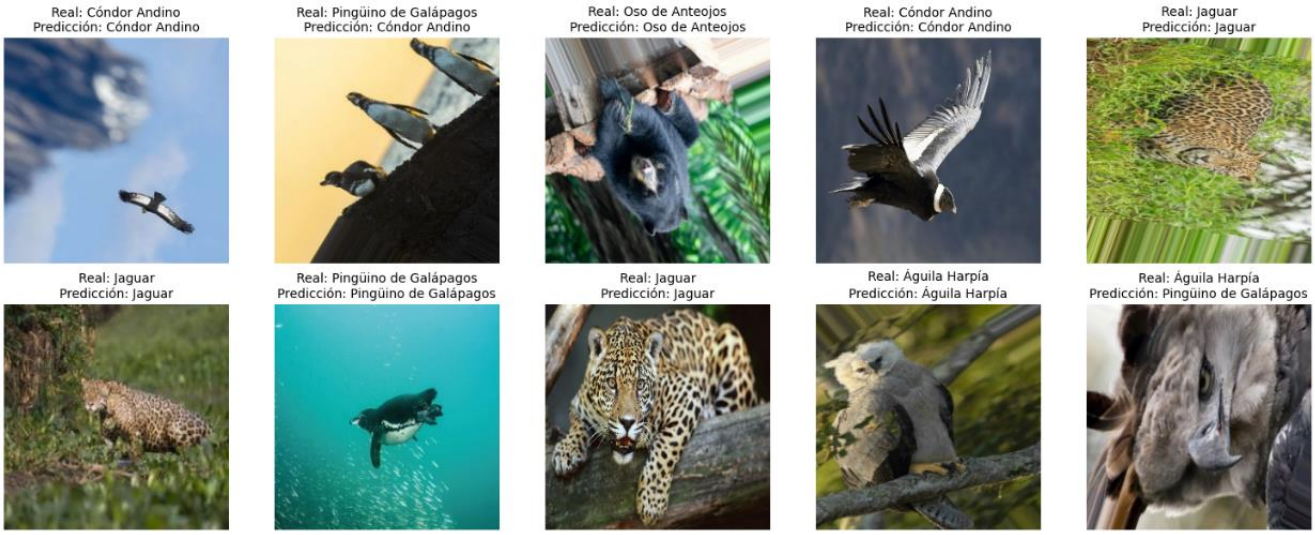
```
# Hacer predicciones con el modelo
predicciones_batch = modelo_batch.predict(X_test)

# Obtener las clases predichas para cada muestra
clases_predichas_batch = np.argmax(predicciones_batch, axis=1)
```

```
48/48 [==============================] - 1s 16ms/step
```

22 Realizar predicciones y obtener valores discretos

Figura 21

Comparación de rendimiento en entrenamiento y validación de modelo en grises



Se han presentado algunas de las primeras predicciones contrastando el valor real de la etiqueta con el predicho por el modelo, aunque no ha acertado a todas las presentadas, la precisión en esta visualización alcanzó el 80% de animales clasificados correctamente. A simple viste se puede notar una confusión entre aves, sin embargo, los valores exactos se manifestarán en la matriz de confusión y las métricas de evaluación general y para cada una de las clases.

3.5.3. Análisis de métricas de evaluación

Como se estableció anteriormente las métricas de evaluación a ser consideradas para establecer una conclusión sobre el rendimiento del modelo son precisión, recall, f1-score, y matriz de confusión.

Modelo con mejor rendimiento entrenado con conjunto en un solo canal de color:

Clase	Precisión	Recall	F1-Score	Soporte
Pingüino de Galápagos	0.48	0.78	0.59	325
Oso de Anteojos	0.86	0.44	0.58	303
Águila Harpía	0.51	0.55	0.53	311
Cóndor Andino	0.57	0.46	0.51	297
Jaguar	0.93	0.87	0.90	299

Tabla No. 1 Métricas de evaluación para el modelo escala de grises

El soporte se refiere a la cantidad de elementos reales por clase que comprenden su porción dentro del conjunto de pruebas. Teniendo como métricas de evaluación para para cada clase del mejor modelo en escala de grises, lo siguiente:

Precisión: La precisión para los mamíferos es destacable a diferencia de la precisión de aves que tiene bastante espacio de mejora con un promedio de 52%. Mientras que mamíferos como el Jaguar lleva la precisión más alta, que demuestra que el 93% de las veces que se ha identificado una especie como clase Jaguar ha sido correcto.

Recall: El Jaguar tiene el puntaje más alto con 87%, sin embargo, el Pingüino de Galápagos tiene el segundo con un recall del 78% que indica que de todas las clases reales en el soporte de esta especie el modelo identificó el 78% de manera correcta.

F1-Score: Como se esperó con los resultados anteriores el Jaguar obtiene un F1-score de 90% lo que muestra un buen rendimiento de la clase al equilibrar la precisión y el recall en valores bastante altos. El más bajo es el Cóndor Andino con 51%.

	Precisión	Recall	F1-Score	Soporte
Precisión			0.62	1535
Promedio Macro	0.67	0.62	0.62	1535
Promedio Ponderado	0.67	0.62	0.62	1535

2 Métricas de rendimiento globales del modelo definido en escala de grises

Matriz de confusión de modelo definido en un solo canal de color:

253	2	44	25	1
72	132	56	32	11
87	10	171	39	4
100	8	48	137	4
14	2	16	8	259

- Pingüino de Galápagos: 253 veces fue correctamente identificada, sin embargo, 44 fue clasificada como Cóndor y 25 veces como Águila Harpía.
- Oso de Anteojos: 132 veces fue identificado correctamente, 72 veces como un Pingüino de Galápagos, 56 como Cóndor y 32 como Águila Harpía.
- Águila Harpía: 171 fue clasificada correctamente, pero 87 identificada erróneamente como un Pingüino de Galápagos y 39 como un Cóndor Andino.
- Cóndor Andino: 137 veces correctamente identificado, 100 veces como Pingüino de Galápagos y 48 como Oso de Anteojos.
- Jaguar: 259 veces clasificado correctamente, muy superior al resto de clases con su mayor confusión de 16 veces como Águila Harpía.

Métricas de evaluación para mejor modelo definido en tres canales de color:

Clase	Precisión	Recall	F1-Score	Soporte
Pingüino de Galápagos	0.71	0.76	0.73	325
Oso de Anteojos	0.88	0.78	0.83	303
Águila Harpía	0.65	0.70	0.67	311
Cóndor Andino	0.66	0.61	0.63	297
Jaguar	0.90	0.93	0.91	299

3 Métricas de evaluación modelo RGB

Teniendo las métricas de evaluación de cada clase para el mejor modelo a RGB de la manera:

Precisión: Se presenta una mayor precisión en cada una de las clases con valores más aceptables para aves, aumentando un 23% en la proporción de verdaderos entre predicciones positivas para el Pingüino de Galápagos.

Recall: Se obtuvo un mayor porcentaje por especie respecto a escala de grises, aumentando significativamente en Oso de Anteojos con 78%, pero manteniendo al Jaguar superior con 93% de instancias identificadas correctamente.

F1-Score: Se ha conseguido un incremento a valores aceptables de rendimiento global por clase, sin embargo, el Cóndor se estableció inferior con un 63% en su equilibrio entre precisión y recall, y el Jaguar continúa destacando con un 91%.

	Precisión	Recall	F1-Score	Soporte
Precisión			0.76	1535
Promedio Macro	0.76	0.76	0.76	1535
Promedio Ponderado	0.76	0.76	0.76	1535

4 Métricas de rendimiento globales del modelo entrenado RGB

Matriz de confusión:

247	2	20	49	7
13	237	31	15	7
39	16	217	30	9
45	8	56	181	7
5	7	9	1	277

- Pingüino de Galápagos: 247 veces del soporte se clasificó correctamente, con 49 veces erróneamente como Cóndor Andino.
- Oso de Anteojos: 237 veces fue correctamente clasificado, pero 31 veces fue identificado como Águila Harpía.
- Águila Harpía: 217 veces correctamente identificado, pero 39 erróneamente como Pingüino de Galápagos y 30 como Cóndor Andino.
- Cóndor Andino: 181 ocasiones correctamente clasificado con confusiones como Águila Harpía 56 veces y 45 como Pingüino de Galápagos.
- Jaguar: 277 veces correctamente identificado como Jaguar, su mayor confusión como Águila Harpía en 9 ocasiones.

En cuanto al rendimiento general de los modelos podemos resaltar una mejora significativa en todas las métricas individuales y globales en el modelo definido en tres canales de color respecto al definido en escala de grises, con un 76% de precisión en general promediando las clases del modelo, respecto al 62% de precisión del modelo a grises indicando una capacidad superior de clasificar las distintas especies de manera correcta. Esto señala la diferencia en la cantidad de información que el modelo puede recoger de tres canales de color en comparación a uno, el proceso de convolución además se desarrolla distinto al generar operaciones y mapas de características con cada uno de estos canales permitiéndole extraer particularidades adicionales por la información extra que recibe.

4. Conclusiones y recomendaciones

4.1. Conclusiones

Se ha conseguido un desarrollo exitoso del modelo de redes neuronales convolucionales, a través del seguimiento de las fases propuestas por CRISP-DM y la capacidad de retroceder entre estas, esto permitió lograr una mejor comprensión y preparación del conjunto para lograr un modelo con un rendimiento mejorado, resultando estas como las fases más demandantes, pues la comprensión de los datos y su preprocesamiento definirán el éxito del proyecto de minería.

La recolección de datos para las especies representó un desafío que fue abordado por medio de la utilización de la API de Google Custem Search, que, aunque presentó limitaciones de respuesta, requirió solo de aplicación de estrategias iterativas y de variación de enfoques para conseguir alrededor de 1000 imágenes por cada una de las especies, lo que facilitó un conjunto de datos sólido para un entrenamiento adecuado.

Conseguir una preparación impecable implicó una alta inversión de tiempo en el proceso de selección de datos de calidad, que se optimizó a través de widgets interactivos para facilitar la visualización y eliminación de datos irrelevantes. El éxito de este proceso puede compromenter el rendimiento del modelo, por lo que se realizaron varias iteraciones. A través de técnicas de aumentación de datos se obtuvieron 2000 imágenes por clase, aplicadas las transformaciones necesarias de dimensión y normalización identificadas previamente. Este enfoque integral resultó en un conjunto de datos robusto y de alto valor para un desarrollo productivo.

La construcción de los modelos presentó una oportunidad de experimentación, pues, aunque se establecieron las funciones de activación y pérdida dada la naturaleza del proyecto de minería de clasificación multiclase, parámetros como el número de capas y filtros, además de la implementación gradual de técnicas de regularización para evitar el sobreajuste, fueron variando e integrándose siendo ajustadas tras las métricas obtenidas durante el entrenamiento de cada modelo previo. Además, el uso de la GPU A100 presentó un rendimiento más acelerado respecto a otras GPUs o TPUs.

Las métricas de evaluación establecidas para medir el rendimiento de los modelos fueron la precisión, recall, f1-score y la matriz de confusión para los que se obtuvo que el modelo en tres canales de color obtuvo un rendimiento general superior al de escala de grises debido a la

cantidad extra de información. Además, del rendimiento de cada clase se obtuvo uno excepcional para los mamíferos como Jaguar y Oso de Anteojos, sin embargo, para aves la precisión no fue tan destacable y generó bastante confusión entre las especies. Esto puede deberse a la diferencia notable de particularidades entre las clases.

El modelo construido es capaz de reconocer y clasificar las especies emblemáticas del Ecuador generalmente, se evidenció una mejora significativa en el rendimiento general en el modelo en tres canales de color con una precisión de 76%, superando el umbral mínimo de precisión definido para considerar al modelo aceptable.Tras el análisis de la precisión del modelo en escala de grises en 62%, se puede inferir que la diferencia de información entre el color y escala de grises influye considerablemente en la extracción de características relevantes.

4.2. Recomendaciones

Con fin de obtener mejores resultados en la precisión del modelo, es imperativo contar con un conjunto extenso de datos de calidad, por ello, extraer una colección de imágenes más amplia puede contribuir al aprendizaje y una mejor generalización, sobre todo en las especies de aves que es donde se produjo complicación.

Aunque fueron evaluadas, en este proyecto no se aplicaron técnicas de eliminación de ruido o corrección de contraste debido a la visual pérdida de información, estas pueden considerarse si se dispone de un conjunto de datos más grande, así conseguir un incremento en la calidad de la colección de datos y aumentar el rendimiento del modelo.

Con un modelo que cuente con una precisión igual de elevada y balanceada en cada una de sus clases, considerar el despliegue de este a un entorno de producción, como aplicaciones web o móviles, así la herramienta puede estar al alcance de cualquier persona natural o especialistas en el campo o incluso facilitar el modelo para que pueda ser integrado con sistemas externos.

BIBLIOGRAFÍA

Adetunji, J. (2021, Julio 15). *theconversation.com*. Obtenido de https://theconversation.com/la-inteligencia-artificial-nos-ayuda-a-estudiar-la-fauna-de-donana-para-mejorar-su-conservacion-162815

Alonso, J. L. (2022, Junio 15). *incentro.com*. Obtenido de https://www.incentro.com/es-ES/blog/que-es-tensorflow

aws. (2023). *aws.amazon.com*. Obtenido de https://aws.amazon.com/es/what-is/python/

Bowen, M. (2023, Junio 1). *intelligentcio.com*. Obtenido de https://www.intelligentcio.com/latam-es/2023/06/01/sas-seeks-crowd-driven-ai-to-protect-endangered-sea-turtles-in-galapagos/

Cloud, I. (2024). *ionos.es*. Obtenido de https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/que-es-keras/

Developers, G. f. (2023, Diciembre 18). *developers.google.com*. Obtenido de https://developers.google.com/custom-search/v1/overview?hl=es-419

Epitech. (2021, julio 08). Obtenido de https://www.epitech-it.es/flask-python/

Google. (2024). *research.google.com*. Obtenido de https://research.google.com/colaboratory/intl/es/faq.html#:~:text=Colab%20es%20un%20servicio%20alojado,de%20datos%20y%20la%20educaci%C3%B3n.

IBM. (2021, 08 17). Obtenido de https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview

IBM. (2021, Agosto 17). *ibm.com*. Obtenido de https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview

IBM. (2023). *ibm.com*. Obtenido de https://www.ibm.com/es-es/topics/convolutional-neural-networks

IBM. (2024). Obtenido de https://www.ibm.com/mx-es/topics/exploratory-data-analysis

lisdatasolutions. (s.f.). Obtenido de https://www.lisdatasolutions.com/es/blog/deep-learning-clasificando-imagenes-con-redes-neuronales/

Microsoft. (2024). Obtenido de https://visualstudio.microsoft.com/es/

GLOSARIO DE TÉRMINOS

Redes neuronales convolucionales (CNN): Técnica de Deep Learning que se especializa en el reconocimiento de imágenes y video.

Normalización: El proceso de ajustar los valores de píxeles de una imagen a una escala común, generalmente entre 0 y 1.

OneHotEncoding: Técnica que representa cada categoría como un vector binario con un solo 1 en la posición correspondiente a esa categoría y 0 en todas las demás posiciones.

Regularización: métodos utilizados para evitar que el modelo se sobreajuste. Incluye técnicas como regularización L2, Drop Out, L1.

Filtro o kernel: Matriz pequeña que se utiliza en operaciones de convolución para identificar características locales como bordes, texturas y patrones en una imagen.

Tensorboard: Herramienta gráfica de TensorFlow que permite la monitorización del rendimiento de los modelos durante y después de entrenarse.

Zancada o Stride: Número de pixeles que se traslada el filtro durante el pooling o agrupación, durante el proceso de convolución.

Pooling o Agrupación: Operación de reducción de dimensión de las imágenes, con fin de extraer las características abstractas más relevantes.

Batch Normalization: Técnica de normalización de las entradas de las capas intermedias que aceleran y estabilizan el entrenamiento del modelo.

DropOut: Técnica de regularización de omite neuronas en un porcentaje definido de manera aleatoria para que el modelo sea capaz de generalizar mejor.

ANEXOS

1. **Enlace a los cuadernos Google Colab**

    a. Modelo CNN definido en RGB

https://colab.research.google.com/drive/1fVRx3-y2OrezxZ6394uQtOJ50VdLmV9M?usp=sharing

    b. Modelo CNN definido en escala de grises

https://colab.research.google.com/drive/1bIjAcv6xwQPZQcCvyKNzpU9bS_sN8sl5?usp=sharing

2. **Enlace Drive a conjuntos de imágenes**

    a. Conjunto de imágenes sin procesar

https://drive.google.com/drive/folders/1dgIjk__wTnBhPu0yY5EljF3eNSETwqmN?usp=sharing

    b. Conjunto de imágenes en escala de grises procesadas

https://drive.google.com/drive/folders/1LKdBPmalWVswJ_YK6Cbe5UryQXo_J_Ib?usp=sharing

    c. Conjunto de imágenes en RGB procesadas

https://drive.google.com/drive/folders/1eIyRD0cwS5x3-RgSFJ2qhlmTRJL6o4LJ?usp=sharing

3. **Enlace Drive modelos CNN entrenados en formato H5**

    a. Modelo CNN en RGB (modelo_batch_RGB.h5)

https://drive.google.com/file/d/18BES9TgWDcYMMfvUHRb5XapNAW3BkM6H/view?usp=sharing

    b. Modelo CNN en escala de grises (modelo_batch.h5)

https://drive.google.com/file/d/1pNmuOMcKGBgtp-ow4LxmpOOH6syQN6oJ/view?usp=sharing