Hecho por:

Andrés Peña
Daniel Quiroz
Luis Torres

Contexto de aplicación

La clasificación de especies de ballenas mediante imágenes forma parte de las aplicaciones de la inteligencia artificial en el ámbito de la **investigación biológica**, la **gestión ambiental** y la **protección de especies en peligro de extinción**. Tradicionalmente, la identificación de ballenas ha requerido la participación de expertos marinos realizando observaciones directas o analizando manualmente grandes volúmenes de fotografías. Este proceso es costoso, demorado y susceptible a errores humanos.

La automatización de la clasificación mediante modelos de visión por computadora permite analizar enormes cantidades de datos de manera eficiente y precisa, optimizando el tiempo y los recursos disponibles. Además, esta tecnología facilita el monitoreo continuo de las poblaciones de ballenas, contribuyendo a detectar cambios en sus patrones migratorios, áreas de alimentación y reproducción, así como también a identificar amenazas emergentes derivadas del cambio climático, la contaminación o la actividad humana.

Objetivo de Machine Learning

Predecir la especie de una ballena a partir de una imagen proporcionada. Se trata de un problema de clasificación supervisada de imágenes, donde a cada imagen de entrada le corresponde una etiqueta que representa su especie.

Dataset

Tipo de datos: Imágenes en formato JPEG con tamaño entre 170KB y 1KB .

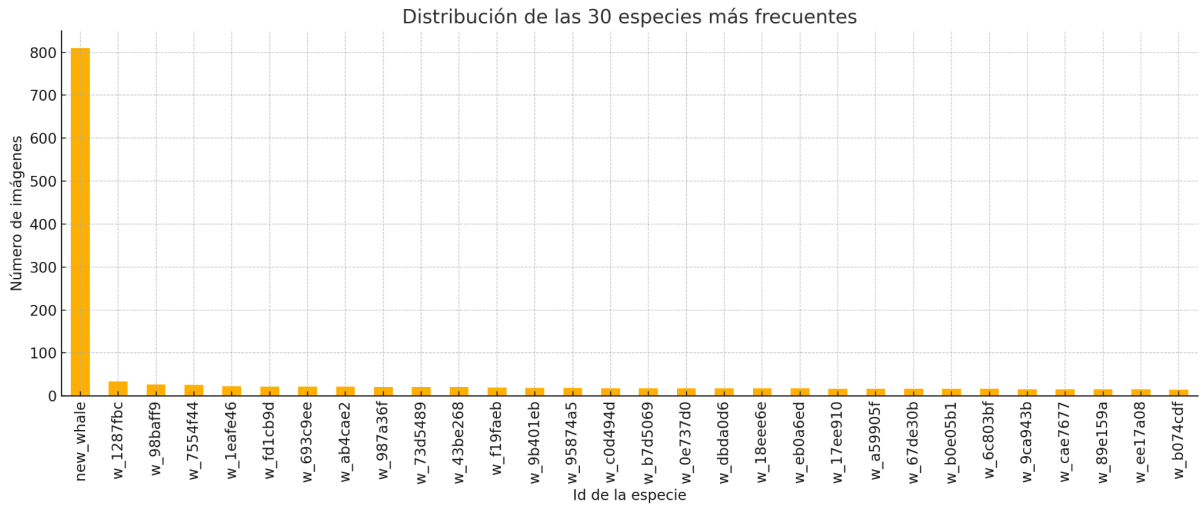
Tamaño del dataset: Cantidad de imágenes 9850, con un tamaño en disco de 289MB

Distribución de clases: La variable objetivo del conjunto de datos (Id) representa las diferentes especies de ballenas. Al analizar la distribución de clases, se observa que las imágenes no están distribuidas de manera equilibrada entre las distintas especies.

- Algunas especies cuentan con varios cientos de imágenes, mientras que otras apenas tienen unas pocas instancias.

- En el análisis inicial, las 30 especies más frecuentes concentran un número significativamente mayor de imágenes en comparación con el resto.

- Además, se identifican múltiples especies que poseen menos de 10 imágenes en el conjunto de entrenamiento.



Métricas de desempeño de machine learning

Exactitud (Accuracy): define la proporción de predicciones correctas respecto al total de muestras evaluadas.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Recall (Sensibilidad o Exhaustividad): Indica qué proporción de los casos positivos reales fueron correctamente identificados.

$$Recall = \frac{TP}{TP+FN}$$

Precision (Precisión): Indica qué proporción de las predicciones positivas fueron correctas.

$$Precisión = \frac{TP}{TP+FP}$$

F1-Score (Media armónica de Precision y Recall): Mide el equilibrio entre Precision y Recall. Es útil en datasets desbalanceados.

$$F1 - Score = 2 \times \frac{Precisión \times Recall}{Precisión + Recall}$$

Balanced Accuracy (Exactitud Balanceada): Es decir, el promedio del Recall de la clase positiva y la clase negativa.

$$Balanced\ Accuracy = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

Donde:

- **TP**: Verdaderos Positivos

- **TN**: Verdaderos Negativos
- **FP**: Falsos Positivos
- **FN**: Falsos Negativos

Métricas de desempeño del negocio

Tasa de Detección Correcta de Especies: Proporción de especies de ballenas correctamente identificadas frente al total de especies evaluadas.
Relacionada directamente con el **Recall macro** (promedio de recalls por clase).

$$Tasa\ de\ Deteccion = \frac{textEspeciescorrectamenteidentificadas}{textTotaldeespecies}$$

Tasa de Error de Clasificación: Porcentaje de especies mal clasificadas, lo cual puede derivar en **decisiones erróneas** en programas de conservación o monitoreo.

$$Error\ de\ Clasificacioon = 1 - Accuracy$$

01 - Exploración de datos

Este notebook presenta los gráficos de distribución e informacion detallada del dataset

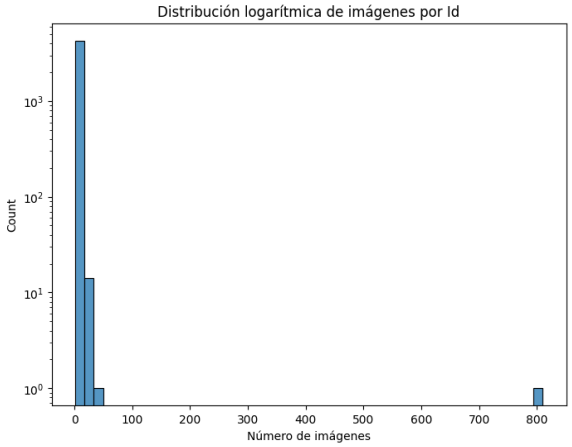


Figura 1. Distribución logarítmica del número de imágenes por clase, evidenciando el fuerte desbalance en el dataset.

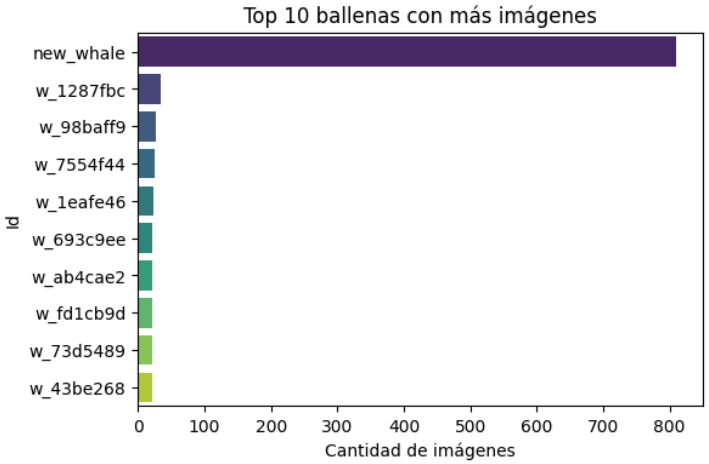
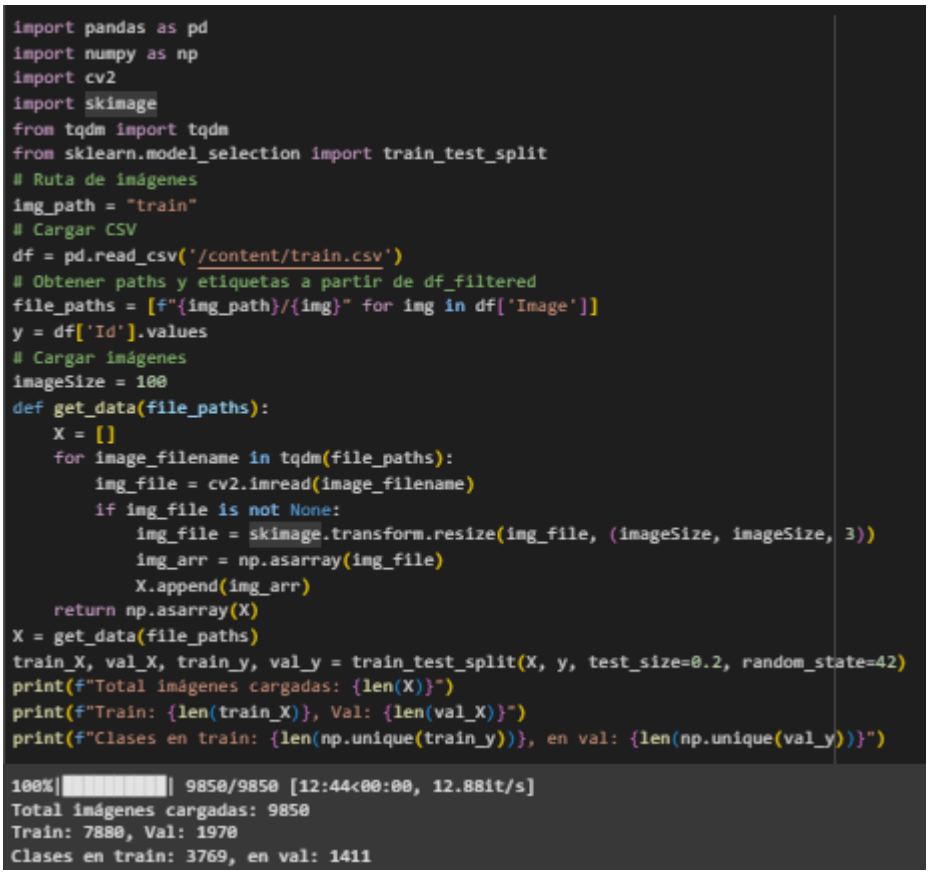


Figura 2. . Distribución de las 10 clases de ballenas con mayor número de imágenes en el conjunto de datos

02 - Preprocesado

En este notebook se realizaron las siguientes acciones:

- Todas las imágenes fueron redimensionadas a 100×100 píxeles y convertidas a arreglos NumPy para su posterior procesamiento por el modelo.
- Se dividió el conjunto de datos en entrenamiento (80 %) y validación (20 %) utilizando train_test_split, sin aplicar estratificación, debido a la elevada cantidad de clases con muy pocas muestras, lo que dificulta una separación representativa.
- Finalmente, se presentó un resumen con el número total de imágenes cargadas, la distribución entre los subconjuntos de entrenamiento y validación, y la cantidad de clases únicas presentes en cada uno..



```
import pandas as pd
import numpy as np
import cv2
import skimage
from tqdm import tqdm
from sklearn.model_selection import train_test_split
# Ruta de imágenes
img_path = "train"
# Cargar CSV
df = pd.read_csv('/content/train.csv')
# Obtener paths y etiquetas a partir de df_filtered
file_paths = [f"{img_path}/{img}" for img in df['Image']]
y = df['Id'].values
# Cargar imágenes
imageSize = 100
def get_data(file_paths):
    X = []
    for image_filename in tqdm(file_paths):
        img_file = cv2.imread(image_filename)
        if img_file is not None:
            img_file = skimage.transform.resize(img_file, (imageSize, imageSize, 3))
            img_arr = np.asarray(img_file)
            X.append(img_arr)
    return np.asarray(X)
X = get_data(file_paths)
train_X, val_X, train_y, val_y = train_test_split(X, y, test_size=0.2, random_state=42)
print(f"Total imágenes cargadas: {len(X)}")
print(f"Train: {len(train_X)}, Val: {len(val_X)}")
print(f"Clases en train: {len(np.unique(train_y))}, en val: {len(np.unique(val_y))}")
```

```
100%|██████████| 9850/9850 [12:44<00:00, 12.88it/s]
Total imágenes cargadas: 9850
Train: 7880, Val: 1970
Clases en train: 3769, en val: 1411
```

Figura 3. Train- test split y conteo de clases por grupo de datos.

03 – Arquitectura de línea base

En este notebook se encuentra el código para llevar a cabo las siguientes tareas:

- Lectura del archivo CSV que contiene las rutas y etiquetas, seguida de la carga de las imágenes correspondientes en memoria.
- Construcción de un modelo de red neuronal convolucional (CNN) compuesto por tres bloques secuenciales de capas Conv2D, MaxPooling2D y Dropout, diseñados para extraer características espaciales y reducir el sobreajuste.
- Compilación del modelo utilizando el optimizador Adam y la función de pérdida categorical_crossentropy, adecuada para problemas de clasificación multiclase.
- Entrenamiento del modelo durante 30 épocas sobre el conjunto de datos previamente dividido en entrenamiento y validación.
- Evaluación mediante accuracy, classification_report, y MAP@5

Descripción de la solución

La solución implementada consiste en un modelo de red neuronal convolucional (CNN) sencillo, compuesto por dos bloques convolucionales con normalización, activaciones ReLU y capas de pooling, seguido de una capa densa intermedia y una capa de salida con softmax para clasificación multiclase. El modelo fue entrenado durante 30 épocas con lotes de 128

imágenes, sobre un conjunto de datos altamente desbalanceado con 4251 clases. La arquitectura y los parámetros fueron diseñados como línea base para validar el pipeline de procesamiento y evaluar la dificultad del problema.

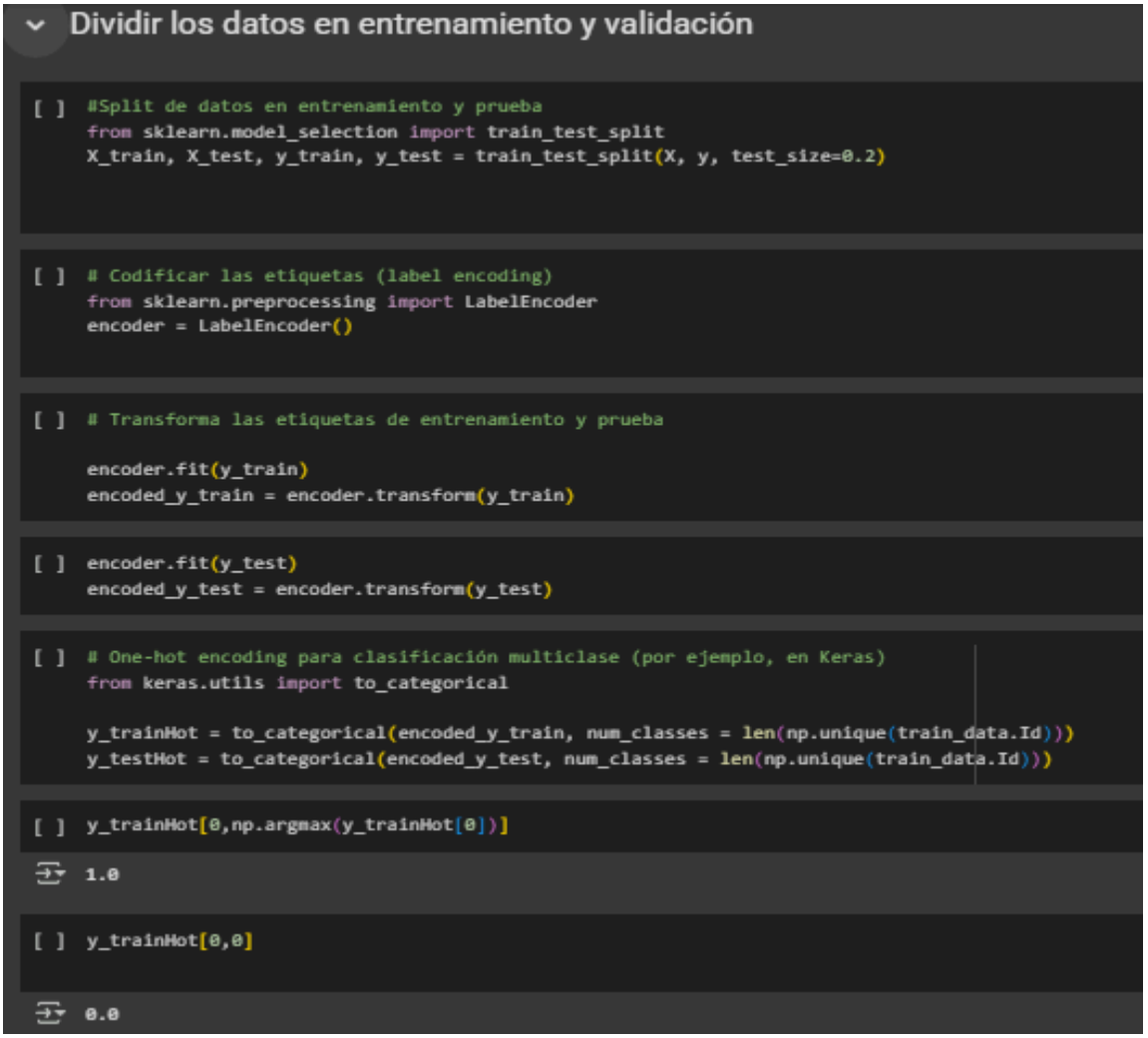


Figura 4. Train- test split y conteo de clases por grupo de datos

Modelo CNN secuencial:

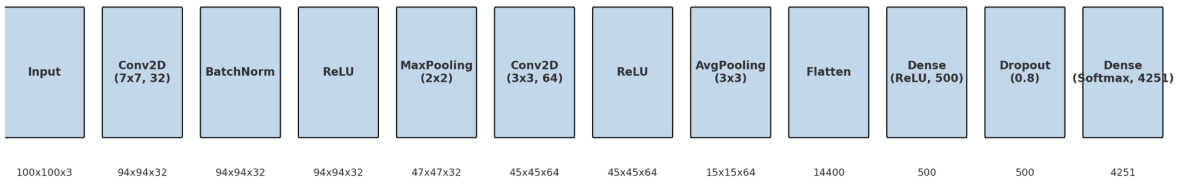


Figura 5. Diagrama de Arquitectura propuesta.

```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv0 (Conv2D)	(None, 94, 94, 32)	4,736
bn0 (BatchNormalization)	(None, 94, 94, 32)	128
activation_4 (Activation)	(None, 94, 94, 32)	0
max_pool (MaxPooling2D)	(None, 47, 47, 32)	0
conv1 (Conv2D)	(None, 45, 45, 64)	18,496
activation_5 (Activation)	(None, 45, 45, 64)	0
avg_pool (AveragePooling2D)	(None, 15, 15, 64)	0
flatten_2 (Flatten)	(None, 14400)	0
rl (Dense)	(None, 500)	7,200,500
dropout_2 (Dropout)	(None, 500)	0
sm (Dense)	(None, 4251)	2,129,751

Total params: 9,353,611 (35.68 MB)
Trainable params: 9,353,547 (35.68 MB)
Non-trainable params: 64 (256.00 B)

Figura 6. Resumen de arquitectura propuesta.

Resultados

Las gráficas de precisión y pérdida muestran que el modelo alcanza rápidamente una precisión cercana al 8.6 %, que se mantiene constante durante el entrenamiento, lo cual sugiere que no está aprendiendo representaciones útiles y su rendimiento es similar al azar. Aunque la pérdida en entrenamiento disminuye de forma continua, la pérdida en validación aumenta con oscilaciones marcadas, lo que evidencia un claro sobreajuste. Este comportamiento se explica por el uso de una arquitectura simple frente a un problema de clasificación con más de 4000 clases altamente desbalanceadas, donde muchas clases tienen muy pocas imágenes, lo que limita significativamente la capacidad de generalización del modelo.

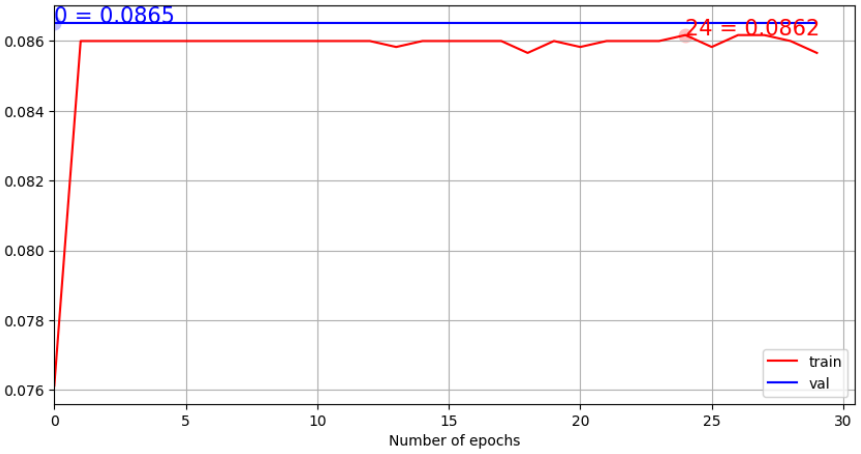


Figura 7.Accuracy del modelo propuesto.

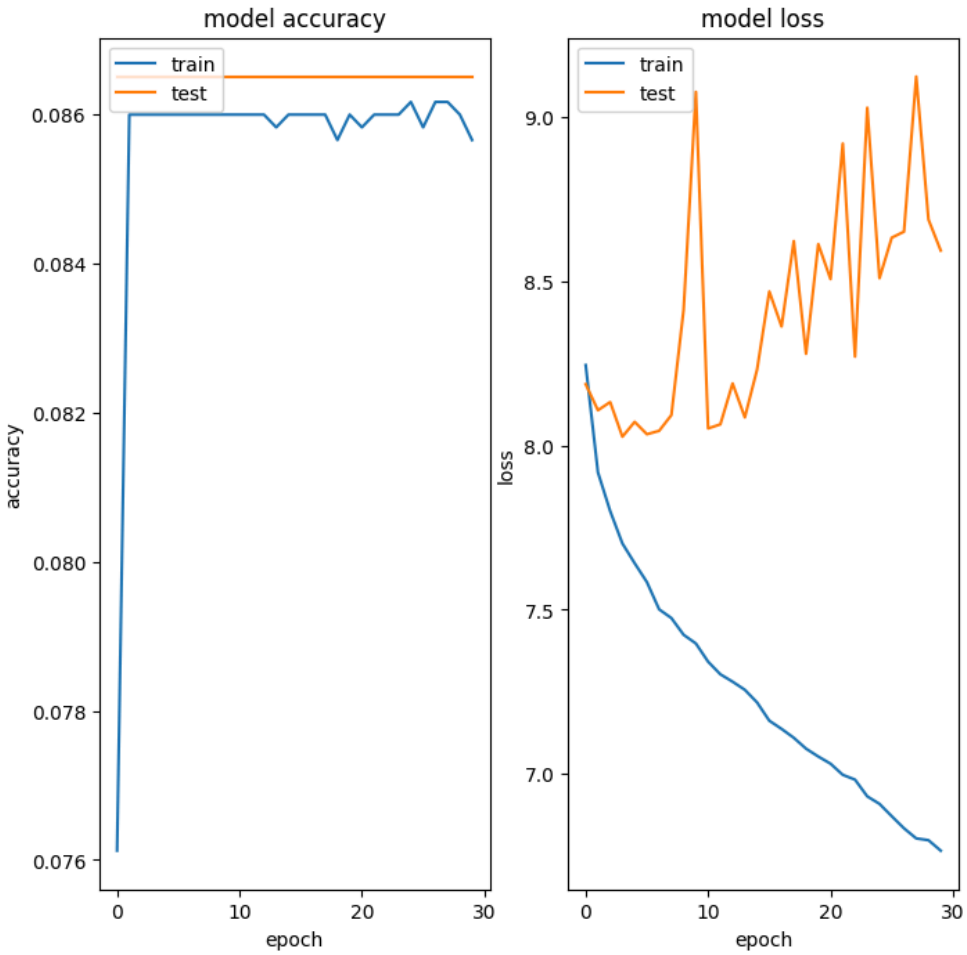


Figura 8.Accuracy y función de pérdida del modelo propuesto.

Conclusiones

Los resultados obtenidos reflejan las limitaciones tanto del conjunto de datos como de la arquitectura empleada. Por un lado, el dataset presenta un severo desbalance de clases, con miles de especies representadas por muy pocas imágenes, lo que impide al modelo aprender patrones representativos para la mayoría de las clases. Por otro lado, se utilizó una arquitectura de red neuronal convolucional sencilla, que no cuenta con la capacidad expresiva necesaria para abordar un problema de clasificación multiclase extrema como este.

Futuras mejoras

Para mejorar el rendimiento del modelo, se propone como trabajo futuro la implementación de estrategias que aborden directamente las limitaciones identificadas. Entre ellas, se incluye el aumento de datos (data augmentation) para mitigar el desbalance entre clases, el diseño de una arquitectura más robusta y profunda que permita extraer representaciones más discriminativas, y la posible reformulación del problema como una tarea de verificación de identidad (retrieval) en lugar de clasificación multiclase directa, lo cual podría ser más adecuado dadas las características del conjunto de datos.

Referencias y resultados previos

- **Fuente principal**:
  - Whale Categorization Playground - Kaggle: https://www.kaggle.com/competitions/whale-categorization-playground
- **Resultados previos**:
  - Modelos de clasificación de imágenes basados en **redes neuronales convolucionales (CNNs)** y **transfer learning** (como ResNet50 o EfficientNet) han mostrado desempeños destacados en problemas similares de clasificación de especies animales.
  - Algunos notebooks públicos en Kaggle aplican **augmentación de datos**, **normalización**, y **optimizadores como Adam** para mejorar la precisión.