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 .

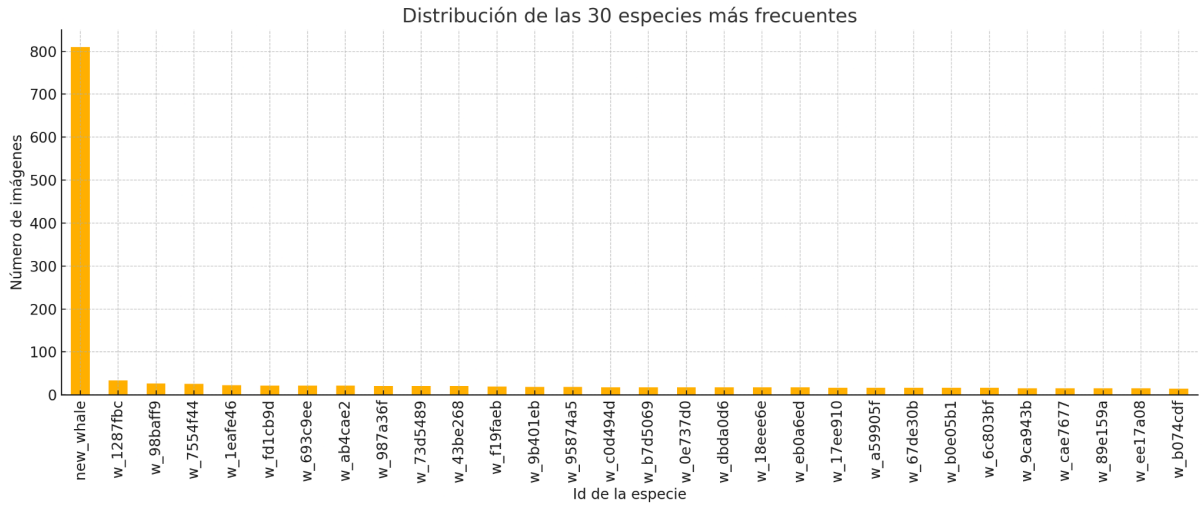
Tamaño del dataset: Cantidad de imágenes 9850, con un tamaño en disco de 289MB

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.

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

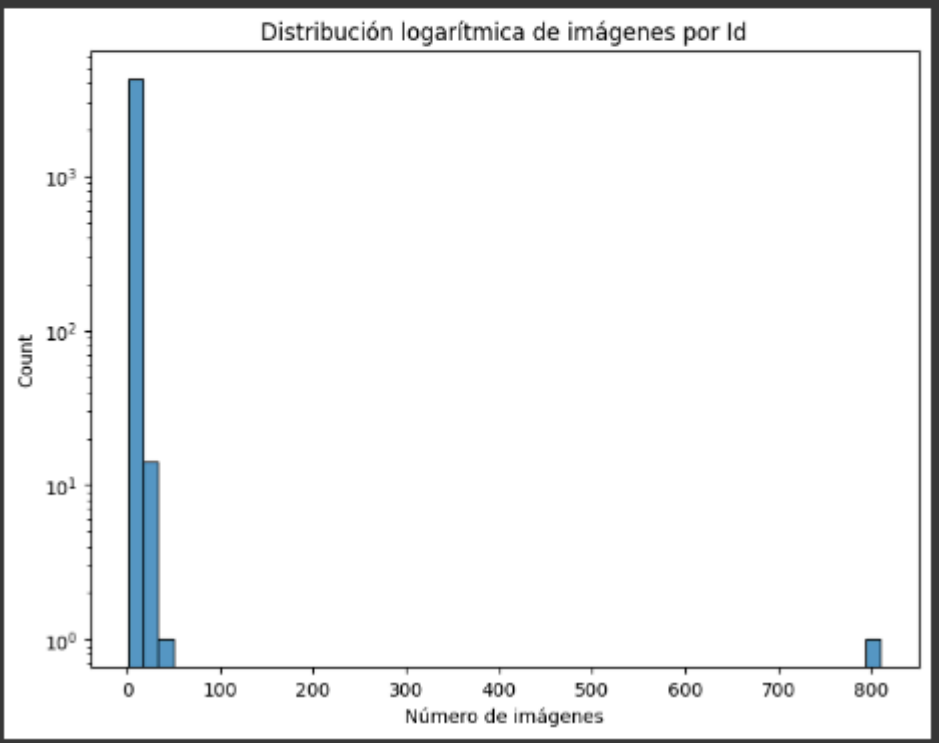
01 - Exploración de datos

Este notebook presenta:

- Gráficos de distribución de clases.
- Ejemplos de imágenes por especie.

Distribución de imágenes por ballena (clase)

```
[ ]  counts = df['Id'].value_counts()
     plt.figure(figsize=(8,6))
     sns.histplot(counts, bins=50)
     plt.yscale('log')
     plt.title('Distribución logarítmica de imágenes por Id')
     plt.xlabel('Número de imágenes')
     plt.show()
```



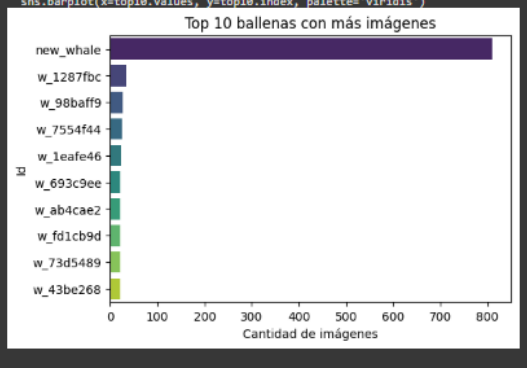
Top 10 ballenas más frecuentes

```
[ ]  top10 = counts.head(10)
     plt.figure(figsize=(6,4))
     sns.barplot(x=top10.values, y=top10.index, palette='viridis')
     plt.title('Top 10 ballenas con más imágenes')
     plt.xlabel('Cantidad de imágenes')
     plt.ylabel('Id')
     plt.show()
```

```
/tmp/ipython-input-9-312246527.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top10.values, y=top10.index, palette='viridis')
```



02 - Preprocesado

En esta etapa se realizaron:

- Se redimensiona todas las imágenes a 100×100 píxeles y las convierte en arrays numpy.
- Aplica train_test_split (80% entrenamiento, 20% validación) sin estratificación, debido a la complejidad de muchas clases con pocas muestras.
- Muestra un resumen con número de imágenes cargadas, distribución entre train/val y cantidad de clases en cada conjunto.

```python
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
```

03 – Arquitectura de línea base

- Lectura del CSV y carga de imágenes
- Construcción de un modelo CNN con 3 bloques Conv2D + MaxPool + Dropout
- Compilación con optimizador Adam
- Entrenamiento sobre 20 épocas
- Evaluación mediante `accuracy`, `classification_report`, y `MAP@5`

Descripción de la solución

Dividir los datos en entrenamiento y validación

```
[ ] #Split de datos en entrenamiento y prueba
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
[ ] # Codificar las etiquetas (label encoding)
    from sklearn.preprocessing import LabelEncoder
    encoder = LabelEncoder()
```

```
[ ] # Transforma las etiquetas de entrenamiento y prueba

    encoder.fit(y_train)
    encoded_y_train = encoder.transform(y_train)
```

```
[ ] encoder.fit(y_test)
    encoded_y_test = encoder.transform(y_test)
```

```
[ ] # One-hot encoding para clasificación multiclase (por ejemplo, en Keras)
    from keras.utils import to_categorical

    y_trainHot = to_categorical(encoded_y_train, num_classes = len(np.unique(train_data.Id)))
    y_testHot = to_categorical(encoded_y_test, num_classes = len(np.unique(train_data.Id)))
```

```
[ ] y_trainHot[0,np.argmax(y_trainHot[0])]
```

```
1.0
```

```
[ ] y_trainHot[0,0]
```

```
0.0
```

Modelo CNN secuencial:

Construir la red convolucional

```
[ ] batch_size = 128
    num_classes = len(np.unique(train_data.Id))
    epochs = 10
    img_rows,img_cols=100,100
    input_shape = (img_rows, img_cols, 3)
    e = 2
```

```
[ ] num_classes
```

```
4251
```

```python
model = Sequential()
model.add(Conv2D(32, (7, 7), strides = (1, 1), name = 'conv0', input_shape = (100, 100, 3)))
model.add(BatchNormalization(axis = 3, name = 'bn0'))
model.add(Activation('relu'))
model.add(MaxPooling2D((2, 2), name='max_pool'))
model.add(Conv2D(64, (3, 3), strides = (1,1), name="conv1"))
model.add(Activation('relu'))
model.add(AveragePooling2D((3, 3), name='avg_pool'))
model.add(Flatten())
model.add(Dense(500, activation="relu", name='rl'))
model.add(Dropout(0.8))
model.add(Dense(num_classes, activation='softmax', name='sm'))
model.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
```

```
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)
```

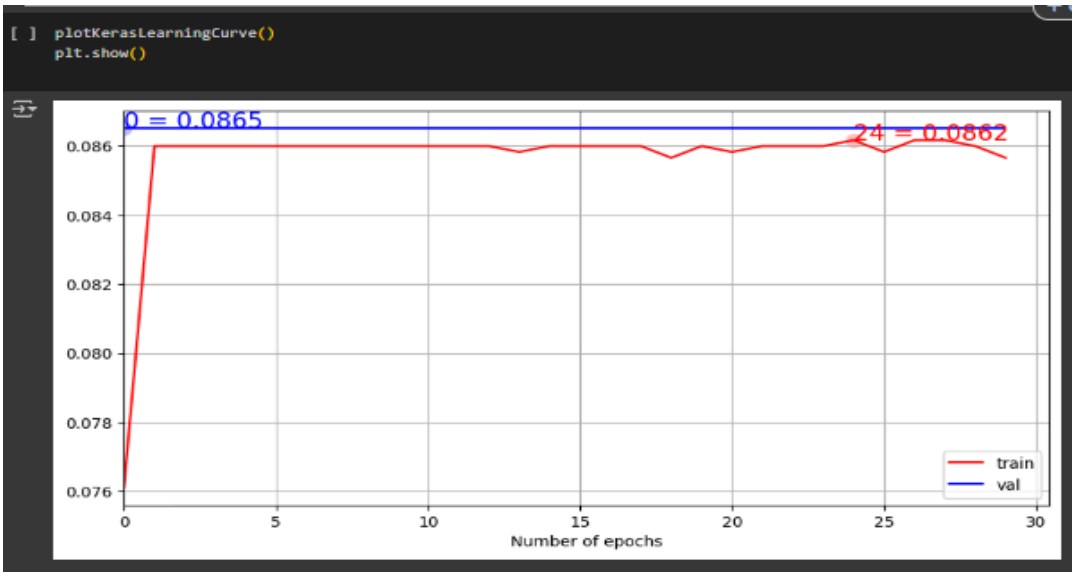
- **Preprocesado**:
    - Imágenes redimensionadas a 100x100x3
    - Normalización entre 0 y 1
    - Codificación categórica de etiquetas
- **Entrenamiento**:
    - 20 épocas
    - EarlyStopping y ReduceLROnPlateau
    - Split 80/20 para validación

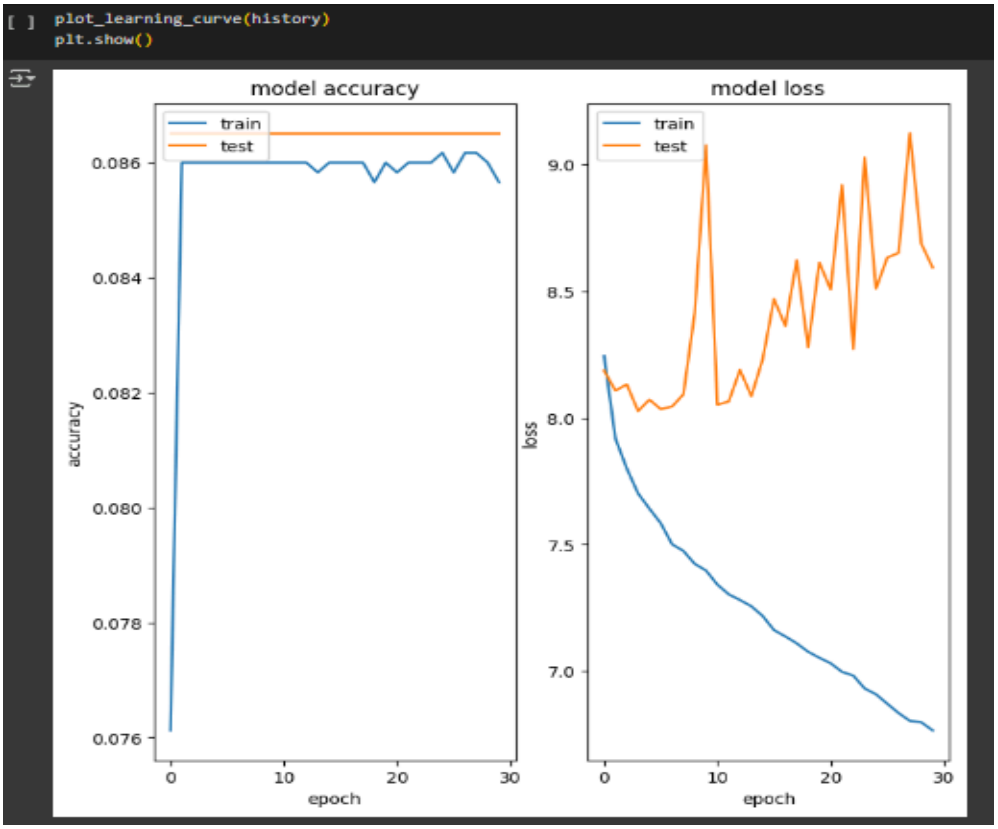
Disponibilidad de datos

- Se utilizó el API de Kaggle con
- Se desempaquetaron las imágenes y se cargaron desde la carpeta `train/`
- En cada notebook se incluye el proceso de descarga:

```
!pip install kaggle
!mkdir -p ~/.kaggle
!cp /content/drive/MyDrive/kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle competitions download -c whale-categorization-playground -p /content/
!unzip /content/whale-categorization-playground.zip -d /content/
```

Resultados





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.