DISEÑO DE UNA RED CONVOLUCIONAL PARA RECONOCIMIENTO DE AVES EN LA UNIVERSIDAD LIBRE DE BOGOTÁ, SEDE EL BOSQUE

Design of a Convolutional Network for Recognition of birds at the Universidad Libre de Bogotá, sede El Bosque Popular

Dayana Benitez Orozco[1]
Henry Steveen Tarazona Tarazona[2]
Claudia Marcela Cifuentes Velásquez[3]

[1]Universidad Libre, Bogotá, Colombia, *dayana-benitezo@unilibre.edu.co*
[2]Universidad Libre, Bogotá, Colombia, *henrys-tarazonat@unilibre.edu.co*
[3]Universidad Libre, Bogotá, Colombia, *claudiam.cifuentesv@unilibre.edu.co*

RESUMEN

El crecimiento urbano acelerado y la expansión de áreas metropolitanas plantean desafíos para la conservación de la biodiversidad, especialmente en regiones ricas en especies. A medida que las ciudades crecen, los hábitats naturales se fragmentan y degradan, lo que puede llevar a la disminución de especies. La actividad académica y la presencia constante de estudiantes en campus universitarios también generan perturbaciones en el hábitat natural. La construcción de infraestructuras y el tráfico humano pueden alterar los comportamientos de las especies. Es crucial encontrar métodos para monitorear y mitigar estos impactos, contribuyendo a la conservación en entornos académicos y urbanos.

Este estudio propone diseñar un prototipo de red neuronal artificial para analizar el comportamiento de dos especies de aves en la Universidad Libre de Bogotá, Sede El Bosque. Estas redes, inspiradas en el cerebro humano, ofrecen un enfoque poderoso para el análisis de datos complejos, identificando patrones de comportamiento valiosos para la conservación. Se presenta la metodología basada en el marco Scrum y los resultados preliminares, que han sido satisfactorios, logrando identificar con precisión las aves estudiadas. Estos hallazgos demuestran el potencial de las redes neuronales para la conservación de la biodiversidad en entornos urbanos, proporcionando datos precisos para diseñar estrategias de conservación efectivas.

Palabras clave: Redes Neuronales, Ecología Aviar, Inteligencia Artificial, Análisis de Datos, Comportamiento Animal.

ABSTRACT

Accelerated urban growth and the expansion of metropolitan areas pose challenges for biodiversity conservation, especially in species-rich regions. As cities grow, natural habitats become fragmented and degraded, which can lead to species declines. Academic activity and the constant presence of students on university campuses also generate disturbances in the natural habitat. The construction of infrastructure and human traffic can alter the behavior of species. It is crucial to find methods to monitor and mitigate these impacts, contributing to conservation in academic and urban environments.

This study proposes to design a prototype of an artificial neural network to analyze the behavior of two species of birds at the Universidad Libre de Bogotá, El Bosque headquarters. These networks, inspired by the human brain, offer a powerful approach to analyzing complex data, identifying behavioral patterns valuable for conservation. The methodology, based on the Scrum framework, and the preliminary results are presented, which have been satisfactory, managing to accurately identify the birds studied. These findings demonstrate the potential of neural networks for biodiversity conservation in urban environments, providing accurate data to design effective conservation strategies.

Keywords: Neural Networks, Avian Ecology, Artificial Intelligence, Data Analysis, Animal Behavior.

1. INTRODUCCIÓN

Bogotá, la capital de Colombia, alberga una de las mayores diversidades de aves en su área urbana. Colombia se destaca como uno de los países con una riqueza natural excepcional, gracias a su ubicación privilegiada en la zona intertropical. Aquí, la diversidad de aves es asombrosa, representando casi el 20% de todas las especies del mundo [1].

Por ende, Colombia ha sido galardonada por tercer año consecutivo en el Global Big Day [1]y como el país con el mayor número de registros de aves, el país registró el mayor número de especies de aves durante el concurso realizado el 13 de mayo del 2023, con 1547 registros superó a Perú que se quedó con el segundo lugar, los registros biológicos de Colombia señalan la existencia de más de 1.920 especies [2].

La Universidad Libre, ubicada contigua al reconocido Jardín Botánico, se erige como un nuevo epicentro de vida natural, en este entorno, las aves encuentran un hábitat idóneo gracias a la gran variedad de plantas y animales que lo habitan, destacándose por su riqueza y diversidad biológica. Este enclave no solo promueve el estudio y la conservación de la naturaleza, sino que también ofrece un espacio vital para la interacción y el aprendizaje, fusionando la academia con el entorno natural de manera armoniosa.

La intensa actividad humana, en particular la de los estudiantes, puede generar complicaciones para la conservación de esta especie y el lugar donde vive. A menudo, la falta de conciencia ambiental de algunos individuos puede llevar a perturbaciones en el entorno, poniendo en riesgo la vida silvestre que lo habita.

La presencia de humanos, especialmente en áreas concurridas como la Universidad Libre y el Jardín Botánico, puede afectar el comportamiento y el hábitat de las aves. Es crucial evaluar cómo estas interacciones pueden influir en la conservación de las especies.

Se utilizaron Redes Neuronales Convolucionales (CNN) porque son especialmente efectivas para procesar y analizar imágenes, permitiendo la extracción automática de características relevantes. Además, las CNN manejan mejor la estructura espacial de las imágenes y requieren menos parámetros que las redes densas.

A diferencia de las Redes Neuronales Recurrentes RNN, que son mejores para datos secuenciales, las CNN sobresalen en el análisis visual, siendo superiores a métodos como Máquinas de Soporte Vectorial SVM o árboles de decisión en la captura de patrones visuales complejos.

Si bien existen métodos tradicionales para estudiar aves, como la observación directa y el análisis de datos de campo, estos pueden ser limitados en términos de eficiencia y alcance.

El uso de una red neuronal artificial permite procesar grandes volúmenes de datos de forma más eficiente y precisa [3], proporcionando información más detallados sobre el comportamiento de las aves.

Comprender el comportamiento y las necesidades de las aves en entornos urbanos es esencial para desarrollar planes de conservación y gestión de ecosistemas en Bogotá. Esto permitirá proteger a las especies y promover una coexistencia armoniosa entre aves y humanos.

El diseño de un prototipo de red neuronal artificial para estudiar las aves en Bogotá aborda los desafíos de conservación en el entorno urbano único y biodiverso de la capital colombiana.

Este proyecto no solo busca profundizar en el conocimiento científico sobre las aves, sino también promover el aumento de la conciencia ambiental y la protección de la diversidad biológica en entornos urbanos en Bogotá.

Al entender mejor las necesidades y los desafíos que enfrenta esta especie, se pueden tomar medidas más efectivas para garantizar su supervivencia y preservar el equilibrio ecológico en la ciudad.

De acuerdo con lo dicho anteriormente, se pretende desarrollar una red neuronal destinada al análisis exhaustivo de las aves de la Universidad Libre, que se pueden observar en la Tabla 1, mediante la recopilación selecta de información de fuentes científicas y verificadas.

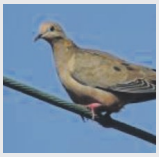
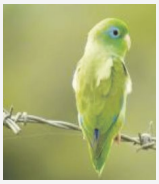
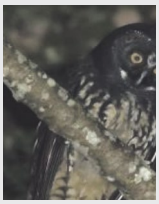
Todo esto orientado hacia un entrenamiento continuo de la red, buscando una constante adquisición de conocimientos respecto a los patrones, comportamientos y rasgos distintivos de las aves.

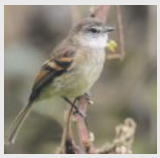
ISSN 2322-8415 / Año 2024 / No. 24 - Vol. 14 / Páginas 38-52 Universidad Libre - Facultad de Ingeniería

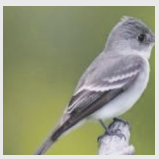
Tabla 1.

Aves del jardín Botánico [4]

Garza blanca	Es una especie de ave pelecaniforme de la familia Ardeidae. Es una de las garzas más ampliamente distribuidas por el mundo.	
Colibrí Rutilante	Es una especie de ave apodiforme de la familia de los colibríes (Trochilidae). Se distribuye por buena parte del oeste y el norte de América del Sur. Es una especie nectarívora.	
Águila Cuaresmera	Es una especie de ave accipitriforme de la familia Accipitridae de pico corto pero afilado, carnívora y su plumaje es blanco con negro y a veces amarillo.	
Garza ganadera	Es una especie de ave pelecaniforme de la familia Ardeidae que vive en todas las zonas tropicales, subtropicales y templadas del planeta. Es la única especie del género Bubulcus, aunque algunos expertos consideran especies separadas a sus dos subespecies. A pesar de sus similitudes en el plumaje con las garcetas del género Egretta está emparentada más cercanamente con las garzas.	
Aguililla caminera	Es una especie de ave accipitriforme de la familia Accipitridae. Es a veces colocado en el género monotípico Rupornis en vez de Buteo. Es autóctona de la Región Neotropical, encontrándo se desde el sur de México hasta el norte de Argentina. Mide aproximadamente 35 cm y pesa alrededor de 295 g. Se alimenta de insectos, pequeños mamíferos y pequeños.	

Tingua azul	Es una especie de ave gruiforme de la familia Rallidae que habita los pantanos y humedales de América. Esta especie se clasificaba antes como Porphyrula martinica, resultando los géneros Porphyrula y Porphyrio sinónimos.	
Paloma montera	Es una especie de ave columbiforme de la familia Columbidae propia de sudamérica.	
Perico de anteojos	Es una especie de ave psitaciforme de la familia Psittacidae propia de Centro y Sudamérica.	
Búho cara oscura	Especie de ave de Cuba, La Española, México, América Central y Sudamérica. Pertenece a la familia Strigidae del orden Strigiformes.	
Garrapatero mayor	Es una especie de ave de la familia Cuculidae que vive en Sudamérica y el sur de América Central.	
Colibrí de Mulsant	Es una especie de ave de la familia Trochilidae, que se encuentra en Bolivia, Colombia, Ecuador, y Perú.	
Mosquero Elenia de Montaña	Es una especie de ave de la familia Tyrannidae.	

Mecocerculus leucophrys	Es una especie de ave paseriforme de la familia Tyrannidae.	
Tirano Pirirí	Es una especie de ave de la familia Tyrannidae.	

Papamoscas del Este	Es una especie que integra el género Contopus, de la familia Tyrannidae. Esta ave nidifica en América del Norte, migrando al sur en el otoño, llegando hasta el centro de América del Sur.	

Las redes neuronales convolucionales (CNN) son un tipo especializado de red neuronal creada para identificar y clasificar imágenes que se proporcionan como entrada. Lo que las distingue de otras redes es su capacidad para realizar un procesamiento convolucional específico.

A diferencia de las redes neuronales tradicionales, que pueden necesitar datos preprocesados de las imágenes, las CNN están diseñadas para trabajar directamente con las imágenes y extraer características relevantes de ellas [5].

Cuando se trata de aplicaciones en ingeniería, lo más notable de las redes neuronales convolucionales (CNN) son la arquitectura de red más comúnmente empleada. Investigadores de Google desarrollaron un método para visualizar cómo una red convolucional representa la información internamente.

Este enfoque da lugar a imágenes que evocan el estilo de los cuadros surrealistas, la técnica se conoce como Deep Dream [6], Un ejemplo de una imagen generada con esta técnica se muestra en la Imagen 1.

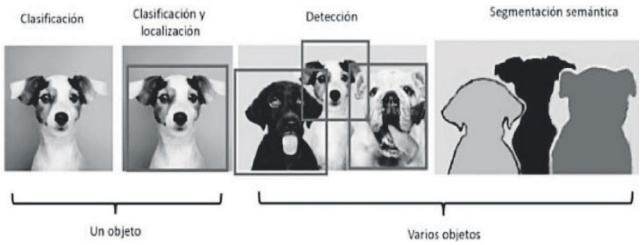


Imagen 1.
Uso de redes neuronales convolucionales para fines artísticos [6].

Otra parte importante en las redes convolucionales es lo que llamamos la capa de agrupación. Esta se encarga de analizar los datos de entrada detectando patrones, texturas, bordes y otras características mediante un filtro seleccionado que recorrerá todos los píxeles de la imagen, generando así una nueva matriz de salida. Estas capas se suelen colocar hacia el final de la red, ya que ayudan a resumir lo más relevante de la imagen para facilitar la toma de decisiones finales.

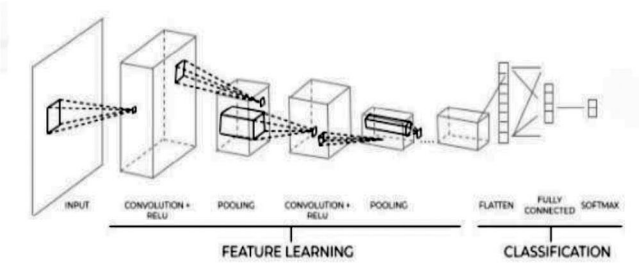


Imagen 1.
Esquema de una Arquitectura CNN [7].

ISSN 2322-8415 / Año 2024 / No. 24 - Vol. 14 / Páginas 38-52 Universidad Libre - Facultad de Ingeniería

Las CNN se destacan por su capacidad para procesar datos estructurados en cuadrícula, como las imágenes. Estas capas utilizan filtros que se desplazan sobre la entrada para generar mapas de características, capturando patrones y texturas específicos. Tras la convolución, se aplica una función de activación, usualmente ReLU, que introduce no linealidades. A menudo, se incluye una etapa de pooling para reducir la dimensionalidad de los datos. Los parámetros de los filtros y los biases se ajustan durante el entrenamiento. Las capas convolucionales son eficaces porque reducen el número de parámetros en comparación con las capas densamente conectadas, mejorando la eficiencia y reduciendo el riesgo de sobreajuste, y son particularmente buenas para capturar la estructura espacial de los datos [8]. En la Imagen 2, observamos cómo la Capa Convolucional procesa los datos de entrada por un filtro, capturando su esencia.

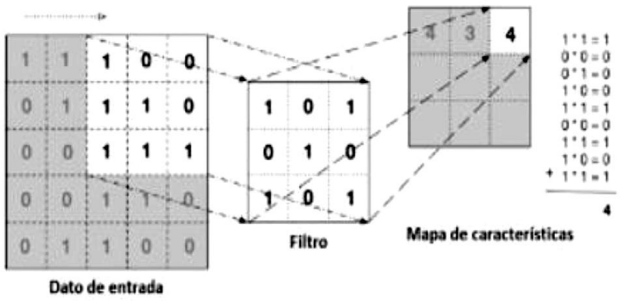


Imagen 2.
Convolución en CNN [8].

Las redes neuronales profundas se componen de varias capas organizadas jerárquicamente. Las capas iniciales detectan características básicas de los datos, y esta información se transmite a capas posteriores que identifican patrones más complejos. Cuando los datos presentan una separación no lineal, se suelen usar al menos tres capas ocultas intermedias. En estas capas, las salidas de una neurona se convierten en las entradas de la siguiente, permitiendo a la red captar relaciones más complejas entre los datos y así abordar problemas más difíciles con mayor precisión [7]. Todo este proceso está ilustrado en la Imagen 3.

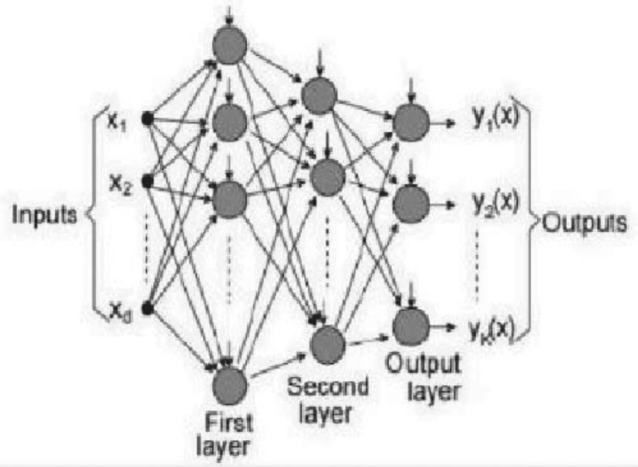


Imagen 3.
Modelo Red Neuronal Multicapa [7]

Durante el proceso de aprendizaje, la red ajusta sus conexiones según cómo responde a los estímulos recibidos. Posteriormente, entra en juego la etapa de recuperación, donde la red emplea lo aprendido para responder de forma específica a nuevas entradas, centrándose en el aprendizaje supervisado al predecir una categoría dentro de un conjunto de datos tras haber sido previamente entrenada [8].

En el periodo de entrenamiento, el algoritmo se familiariza con una serie de ejemplos y resultados deseados asociados a cada uno. El propósito es desarrollar un algoritmo capaz de predecir con precisión el resultado para una nueva entrada, basándose en lo aprendido de los ejemplos durante el entrenamiento; así, el sistema debe reconocer los patrones presentes en las muestras y clasificar las nuevas entradas en función de ellos [8].

El estudio de la avifauna en entornos universitarios es fundamental, dado que la

interacción entre las aves y la comunidad universitaria, compuesta por alumnos, docentes y empleados administrativos de la Universidad Libre, es muy frecuente debido al uso compartido del mismo espacio. Actualmente, las aves enfrentan desafíos como competencia por recursos, contaminación, depredadores urbanos y riesgo de colisiones [9]. Sin embargo, los métodos tradicionales de observación y clasificación de aves pueden ser laboriosos y propensos a errores. Por lo tanto, el problema radica en la necesidad de diseñar una red neuronal eficiente que pueda identificar y clasificar diferentes especies de aves a partir de imágenes.

Este sistema debe ser capaz de procesar grandes volúmenes de datos en tiempo real y proporcionar resultados precisos que puedan utilizarse para la investigación y conservación de la avifauna en la Universidad Libre, ya que es de suma importancia realizar el cuidado de nuestras especies porque Colombia es el hogar de más de 1,900 especies de aves, lo que representa alrededor del 20% de las especies de aves del mundo. Esta cifra es impresionante y convierte a Colombia en uno de los países más ricos en avifauna del planeta [1].

2. MATERIALES Y MÉTODOS

La investigación se realizó en el entorno urbano de Bogotá, específicamente en la Universidad Libre y el Jardín Botánico de Bogotá, durante el período de noviembre de 2023 a marzo de 2024.La presente investigación es de tipo descriptiva, enfocada en medir y comprender el fenómeno de la presencia y comportamiento de las aves en el entorno urbano de Bogotá, específicamente en la Universidad Libre y sus alrededores. El objetivo principal es recopilar información detallada sobre la presencia de las aves, sus hábitos, comportamientos y posibles amenazas en un entorno urbano, sin intentar establecer relaciones causales entre las variables.

El diseño de investigación es de campo, ya que los datos se recopilan directamente en el entorno natural de las aves, alrededor de la Universidad Libre y el Jardín Botánico de Bogotá. Los investigadores actúan como observadores imparciales para estudiar el fenómeno en su entorno natural, registrando la presencia de las aves, sus comportamientos, interacciones con el entorno y posibles amenazas. Este enfoque se complementa con un diseño documental que utiliza información de diversas fuentes para enriquecer el estudio.

La investigación utiliza un enfoque mixto, combinando métodos cuantitativos y cualitativos. El enfoque cuantitativo permite medir y cuantificar aspectos como la frecuencia de avistamientos, la distribución geográfica y la relación entre la presencia de las aves y variables ambientales. Por otro lado, el enfoque cualitativo se emplea para comprender las percepciones, actitudes y comportamientos de las personas que interactúan con el entorno de las aves, mediante observaciones participantes y análisis de contenido.

El proyecto se trabajó en 7 fases que se desarrollaron de la siguiente manera:

Fase 1: Importación de librerías y Graficación de Imágenes de Aves

1. Importación de Librerías:

- Iniciar el proyecto importando todas las librerías necesarias para el análisis y procesamiento de imágenes de aves. Esto incluye bibliotecas como ‘Numpy’, ‘Matplotlib’, ’Pandas’, ‘Tensorflow’, ‘Keras’, y ‘Opencv’.

ISSN 2322-8415 / Año 2024 / No. 24 - Vol. 14 / Páginas 38-52 Universidad Libre - Facultad de Ingeniería

- Utilizar 'Matplotlib' para la visualización y 'Opencv` para la manipulación de imágenes.

2. Carga y visualización de imágenes:

- Cargar un conjunto de imágenes de aves desde un directorio específico.
- Utilizar funciones de 'Matplotlib' para graficar y mostrar algunas de estas imágenes, proporcionando una vista preliminar de los datos con los que se trabajará.

Fase 2: Generación de Imágenes con ImageDataGenerator y Separación de Grupos de Entrenamiento y Validación.

1. Preparación del Generador de Datos:

- Utilizar 'ImageDataGenerator' de Keras para crear generadores de datos que aplicarán transformaciones en tiempo real a las imágenes.
- Configurar los generadores para realizar aumentos de datos, como rotaciones, escalados y ajustes de brillo, lo cual ayudará a aumentar la diversidad del conjunto de datos.

2. Separación del conjunto de datos:

- Dividir el conjunto de imágenes en grupos de entrenamiento y validación.
- Asignar el generador de datos correspondiente a cada grupo, asegurando que ambos generadores utilicen transformaciones similares para mantener la consistencia.

Fase 3: Ploteo de Gráficas Creadas por ImageDataGenerator

1. Visualización de Imágenes Aumentadas:

- Generar un lote de imágenes aumentadas utilizando el 'ImageDataGenerator' configurado en la fase anterior.
- Utilizar 'Matplotlib' para graficar y visualizar las imágenes transformadas. Mostrar varias imágenes en una cuadrícula para observar las diversas transformaciones aplicadas.

2. Análisis de aumentación:

- Analizar visualmente las imágenes para verificar que las transformaciones mejoren la diversidad del conjunto de datos sin distorsionar demasiado las características esenciales de las aves.

Fase 4: Diseño de la Arquitectura de la Red Neuronal

1. Definición del Modelo:

- Utilizar la API de Keras para definir la arquitectura de la red neuronal. Esto incluye capas de entrada, capas ocultas (convolucionales y de pooling) y capas de salida.
- Seleccionar una arquitectura adecuada para el reconocimiento de imágenes, como una red convolucional (CNN).

2. Configuración de hiperparámetros:

- Establecer hiperparámetros como el tamaño del lote, el número de épocas, la tasa de aprendizaje y la función de pérdida.
- Seleccionar optimizadores adecuados, como Adam o RMSprop, y funciones de activación como ReLU para las capas ocultas y softmax para la capa de salida.

Fase 5: Entrenamiento de la Red Neuronal por Épocas

1. Compilación del Modelo:

- Compilar el modelo definido en la fase anterior, especificando la función de pérdida, el optimizador y las métricas de evaluación.

2. Entrenamiento del Modelo:

- Utilizar los generadores de datos para entrenar el modelo en lotes.
- Monitorizar el rendimiento del modelo en el conjunto de entrenamiento y validación durante cada época.
- Ajustar los hiperparámetros según sea necesario para mejorar el rendimiento del modelo.

Fase 6: Evaluación y Descripción de las Predicciones

1. Evaluación del modelo:

- Evaluar el modelo entrenado utilizando el conjunto de datos de validación.
- Calcular métricas de rendimiento como la precisión, la sensibilidad y la especificidad.

2. Interpretación de Resultados:

- Analizar los resultados obtenidos para identificar patrones y posibles áreas de mejora.
- Generar un informe descriptivo de las predicciones, destacando los aciertos y errores del modelo.

Fase 7: Predicción y Visualización de la Descripción del Ave

1. Predicción con Nuevos Datos:

- Utilizar el modelo entrenado para realizar predicciones sobre un nuevo conjunto de imágenes de aves no vistas durante el entrenamiento.

2. Visualización de Resultados:

- Mostrar las imágenes junto con las predicciones generadas por la red neuronal.
- Utilizar 'Matplotlib' para graficar las imágenes y superponer las etiquetas predichas, proporcionando una visualización clara de los resultados del modelo.

3. RESULTADOS

Los resultados del proyecto revelan el éxito de la implementación de una red neuronal desarrollada en Google Colab para el estudio y análisis de especies aviares. Esta plataforma colaborativa permitió una eficiente manipulación de datos y un entrenamiento óptimo del modelo, lo que resultó en una precisión notable en la identificación y clasificación de las diferentes especies de aves analizadas.

FASE 1: Importación de librerías y Graficación de Imágenes de Aves.

```
[ ]  import os
     import matplotlib.pyplot as plt
     from tensorflow.keras.preprocessing.image import ImageDataGenerator

     path = "data/"
     for animal in os.listdir(path):
         if not animal.startswith('.'):
           print (animal)
           fig, axes = plt.subplots(1, 5, figsize=(10, 2))
           imagenes = os.listdir(path + animal)[:5]

           for ax, imagen in zip(axes.flat, imagenes):
               img = plt.imread(f"{path}/{animal}/{imagen}")
               ax.imshow(img)
               ax.axis("off")

         plt.tight_layout()
         plt.show()
```

Imagen 4.
Importación de Librerías para la Red Neuronal- Fuente Propia

Imagen 5.
Graficación de las imágenes de Aves. Fuente Propia.

Al observar detenidamente la imagen 4 y 5 se logra evidenciar:

ISSN 2322-8415 / Año 2024 / No. 24 - Vol. 14 / Páginas 38-52 Universidad Libre - Facultad de Ingeniería

- Las librerías necesarias: os, matplotlib.pyplot y tensorflow.keras.preprocessing.image.
- La ruta de la carpeta donde se encuentran los datos (path = "data/").
- La iteración sobre cada uno de los subdirectorios de path que no empiecen por un punto (ignorando los archivos ocultos).
- Cómo imprime el nombre de cada subdirectorio (cada animal).
- La creación de una figura con un conjunto de subgráficos (5 en total) y muestra las primeras 5 imágenes de cada subdirectorio (animal) en cada uno de los subgráficos.

FASE 2: Generación de Imágenes con ImageDataGenerator y separación de grupos de entrenamiento y validación

```
datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    # validation_split=0.3,
    rotation_range=45,
    # width_shift_range=0.5,
    brightness_range=(0.1, 0.9),
    horizontal_flip=True,
    vertical_flip=True,
)


def listar_carpetas_visibles(ruta):
    return [carpeta for carpeta in os.listdir(ruta) if not carpeta.startswith('.')]

train_generator = datagen.flow_from_directory(
    "data",
    target_size=(128, 128),
    batch_size=16,
    class_mode="categorical",
    subset="training",
    classes=listar_carpetas_visibles("data")
)

validation_generator = datagen.flow_from_directory(
    "data",
    target_size=(128, 128),
    batch_size=16,
    class_mode="categorical",
    subset="validation",
    classes=listar_carpetas_visibles("data")
)
```

Imagen 6.
Configuración de aumento de datos. Fuente Propia.

```
Found 35 images belonging to 2 classes.
Found 0 images belonging to 2 classes.
```

Imagen 7.
Resultados de la búsqueda de imágenes. Fuente Propia

En la segunda fase, como se puede apreciar en las imágenes proporcionadas 6 y 7, se logra observar:

Para comenzar, se define la función 'load_data', la cual acepta dos parámetros:

'batch_size', que determina el tamaño de los lotes de datos a procesar, y 'img_height', que especifica la altura deseada para las imágenes. A continuación, se crea un objeto 'ImageDataGenerator' para llevar a cabo operaciones de aumento de datos en las imágenes, permitiendo así diversificar el conjunto de datos de entrenamiento y mejorar la generalización del modelo. Luego, se utiliza el método 'flow_from_directory' del objeto `ImageDataGenerator` para cargar los datos de entrenamiento y validación desde directorios específicos en el disco, facilitando así la organización y gestión de los conjuntos de datos. Finalmente, la función devuelve dos generadores de datos: uno para los datos de entrenamiento y otro para los datos de validación, los cuales serán utilizados en el proceso de entrenamiento y evaluación del modelo, respectivamente.

FASE 3: Ploteo de gráficas creadas por ImageDataGenerator

```
im = next(train_generator)
fig, axes = plt.subplots(1, 2, figsize=(10, 10))

for i, ax in enumerate(axes.flatten()):
    ax.imshow(im[0][i])
```

Imagen 8.
Visualización de imágenes con el generador de datos. Fuente Propia.

En la tercera fase, como se puede apreciar en la imagen 8, se logra observar el siguiente fragmento de código el cual cumple la función de:

im = next(train_generator): Esta línea obtiene la siguiente imagen (o muestra) de datos del generador de entrenamiento.

fig, axes = plt.subplots(1, 2, figsize=(10, 10)): Esta línea crea una figura con dos subplots (es decir, dos áreas para mostrar gráficos) lado a lado, con un tamaño de figura de 10x10 pulgadas.

for i, ax in enumerate (axes.flatten()): Este es un bucle for que itera sobre los subplots y los asigna a la variable ax. La función 'Enumerate' se utiliza para realizar un seguimiento del índice i de cada subplot.

ax.imshow(im[0][i]): Esta línea muestra la imagen actual (almacenada en im) en el subplot actual (almacenado en ax). El índice se utiliza para mostrar la imagen correcta en el subplot correcto.

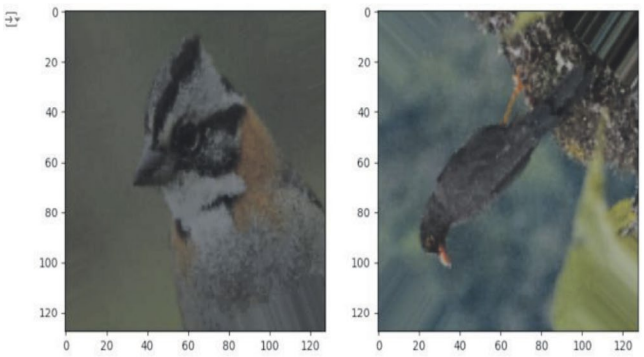


Imagen 9.
Lectura y Escaneo de Datos. Fuente Propia.

im = next (train_generator): Esta línea se repite después del bucle for para obtener la siguiente imagen del generador de entrenamiento y mostrarla en el siguiente lote de subplots.

FASE 4: Diseño de la Arquitectura de la Red Neuronal

En la imagen 10, el código de una red neural convolucional (CNN) usando la API Keras en TensorFlow para clasificar imágenes. Comienza importando librerías necesarias para manejar archivos, manipular datos, procesar imágenes y construir el modelo CNN. El modelo CNN consta de capas convolucionales, max-pooling, dropout, flatten y dense.

```
from glob import glob

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from PIL import Image
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import Convolution2D, Dense, Dropout, Flatten, MaxPooling2D
from tensorflow.keras.models import Sequential

model = Sequential()

model.add(
    Convolution2D(
        16, (6, 6), input_shape=train_generator.image_shape, activation="relu"
    )
)
model.add(MaxPooling2D(pool_size=(2, 2)))
# El Dropout es una capa de regularización que apaga neuronas aleatoriamente en el entrenamiento
# Es una de muchas técnicas utilizadas para reducir el sobreajuste:
model.add(Dropout(0.2))

model.add(Convolution2D(32, (6, 6), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(2, activation="softmax"))

model.compile(
    loss="categorical_crossentropy",
    optimizer=Adam(0.001),
    metrics=["categorical_accuracy"],
)

model.summary()
```

Imagen 10.
Construcción del modelo de aprendizaje profundo con Keras. Fuente Propia.

El modelo se compila con una función de pérdida de entropía cruzada categórica, un optimizador Adam y una métrica de precisión categórica.

Se imprime un resumen del modelo, el cual proporciona una descripción detallada de la arquitectura del modelo, incluyendo el número de parámetros, forma de entrada y forma de salida.

ISSN 2322-8415 / Año 2024 / No. 24 - Vol. 14 / Páginas 38-52 Universidad Libre - Facultad de Ingeniería

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 123, 123, 16)      1744

 max_pooling2d (MaxPooling2  (None, 61, 61, 16)        0
 D)

 dropout (Dropout)           (None, 61, 61, 16)        0

 conv2d_1 (Conv2D)           (None, 56, 56, 32)        18464

 max_pooling2d_1 (MaxPoolin  (None, 28, 28, 32)        0
 g2D)

 dropout_1 (Dropout)         (None, 28, 28, 32)        0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 2)                 50178

=================================================================
Total params: 70386 (274.95 KB)
Trainable params: 70386 (274.95 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Imagen 11.
Arquitectura de la red neuronal. Fuente Propia.

FASE 5: Entrenamiento de la Red Neuronal por Épocas

```
history = model.fit(
    train_generator,
    # validation_data=validation_generator,
    # validation_steps=116 // 16,
    epochs=15
)
```

```
Epoch 1/15
3/3 [==============================] - 4s 359ms/step - loss: 1.6620 - categorical_accuracy: 0.6000
Epoch 2/15
3/3 [==============================] - 2s 429ms/step - loss: 0.9030 - categorical_accuracy: 0.5143
Epoch 3/15
3/3 [==============================] - 2s 452ms/step - loss: 0.6814 - categorical_accuracy: 0.5714
Epoch 4/15
3/3 [==============================] - 1s 220ms/step - loss: 0.6818 - categorical_accuracy: 0.5714
Epoch 5/15
3/3 [==============================] - 1s 212ms/step - loss: 0.6911 - categorical_accuracy: 0.5714
Epoch 6/15
3/3 [==============================] - 1s 478ms/step - loss: 0.6838 - categorical_accuracy: 0.5714
Epoch 7/15
3/3 [==============================] - 2s 579ms/step - loss: 0.6984 - categorical_accuracy: 0.5714
Epoch 8/15
3/3 [==============================] - 3s 1s/step - loss: 0.6872 - categorical_accuracy: 0.5429
Epoch 9/15
3/3 [==============================] - 1s 224ms/step - loss: 0.6916 - categorical_accuracy: 0.5714
Epoch 10/15
3/3 [==============================] - 1s 488ms/step - loss: 0.7104 - categorical_accuracy: 0.4286
Epoch 11/15
3/3 [==============================] - 1s 213ms/step - loss: 0.7060 - categorical_accuracy: 0.4286
Epoch 12/15
3/3 [==============================] - 1s 230ms/step - loss: 0.6818 - categorical_accuracy: 0.6857
Epoch 13/15
3/3 [==============================] - 1s 255ms/step - loss: 0.6682 - categorical_accuracy: 0.5714
Epoch 14/15
3/3 [==============================] - 1s 233ms/step - loss: 0.6843 - categorical_accuracy: 0.5714
Epoch 15/15
3/3 [==============================] - 1s 224ms/step - loss: 0.6769 - categorical_accuracy: 0.5714
```

Imagen 12.
Registro de entrenamiento de la red neuronal. Fuente Propia.

El código entrena un modelo de red neuronal para clasificar especies de aves, usando el generador de datos train_generator. La función model.fit itera sobre el conjunto de datos durante 15 épocas, imprimiendo la pérdida y la precisión categórica después de cada época. La pérdida disminuye y la precisión categórica aumenta a medida que el modelo mejora su capacidad de clasificación.

FASE 6: Evaluación y Descripción de las Predicciones

```
[ ] classes = list(train_generator.class_indices.keys())
    def predict(path):
        im = np.array(image.load_img(path, target_size=(128, 128)))
        prediction = model.predict(im.reshape(1, 128, 128, 3))

        plt.imshow(im)
        plt.axis("off")
        plt.show()

        pd.Series(prediction[0], index=classes).plot(kind="bar", rot=0)
        plt.show()
        show_description(prediction)


    def show_description(prediction):
      predicted_class_index = np.argmax(prediction)
      predicted_class = classes[predicted_class_index]

      class_descriptions = {
            "mirla": "La paraulata morera, mirla patinaranja, o mirlo (Turdus fuscater) es una especie de ave p
            "copeton": "El chingolo, chincol o copetón (Zonotrichia capensis) es una especie de ave paseriforme
        }
        # Mostrar la descripción de la clase predicha
      if predicted_class in class_descriptions:
          print("Descripción de la clase predicha:", class_descriptions[predicted_class])
      else:
          print("No hay descripción disponible para la clase predicha.")
      # Diccionario de clases y descripciones
```

Imagen 13.
Clasificación de imágenes. Fuente Propia.

La función predict toma como entrada un path (ruta de acceso a una imagen) y realiza las siguientes acciones:

Carga la imagen desde el path proporcionado y la convierte en un array de numpy con una forma de (128, 128, 3).

Utiliza el modelo entrenado model para realizar una predicción sobre la imagen cargada. La predicción se almacena en la variable prediction.

Muestra la imagen cargada usando Matplotlib.

Convierte la predicción en una Serie de pandas con las clases como índice y representa gráficamente el resultado.

Llama a la función show_description para mostrar la descripción de la clase predicha.

La función show_description se encarga de encontrar la clase predicha con mayor probabilidad y buscar su descripción en el diccionario class_descriptions. Si encuentra una descripción, la imprime; de lo contrario, imprime un mensaje indicando que no hay descripción disponible para la clase predicha.

FASE 7: Predicción y visualización de la descripción del ave

```
for img in os.listdir("test"):
  if not img.startswith('.'):
    predict("test/" + img)
```

Imagen 14.
Predicción de las Aves de Prueba.
Fuente Propia.

Este fragmento de código (imagen 14) itera a través de todos los archivos en un directorio llamado "test". Omite los archivos que comienzan con un punto (.), que suelen ser archivos ocultos en sistemas basados en Unix. Para cada archivo restante, está llamando a una función llamada predict con la ruta del archivo como argumento.

Imagen 15.
Ave de Prueba para la Predicción.
Fuente Propia

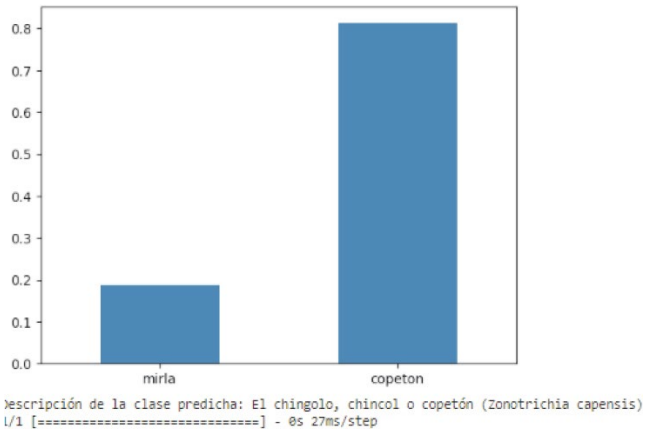


Imagen 16.
Análisis de Datos del Ave de prueba.
Fuente Propia.

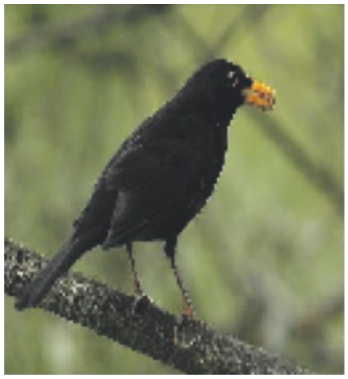


Imagen 17.
Segunda Ave de Prueba para la Predicción.
Fuente Propia

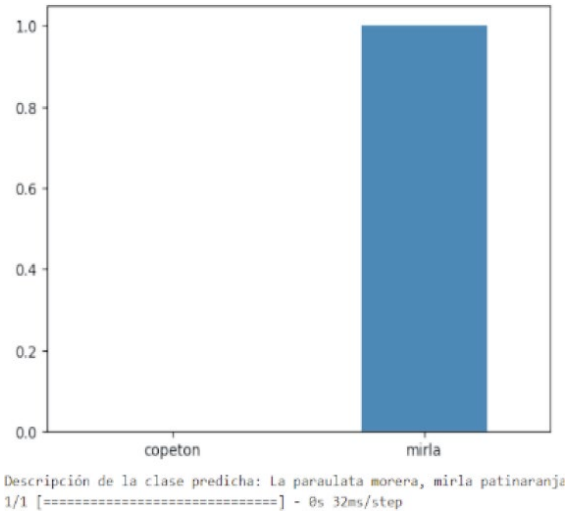


Imagen 18.
Análisis de Datos de la segunda Ave de prueba. Fuente Propia.

ISSN 2322-8415 / Año 2024 / No. 24 - Vol. 14 / Páginas 38-52 Universidad Libre - Facultad de Ingeniería

En la imagen 15,16,17 y 18 se puede observar el resultado de la predicción. En la imagen 15 y 17 se sube la foto de la especie a ser analizada, mientras que la imagen 16 y 18 realiza su predicción afirmando que el ave de muestra cuenta con un 80% de probabilidades de ser un 'Copeton' y 20% de ser una 'Mirla' en el primer ejemplo, y arrojando un 100% de probabilidad en el segundo ejemplo sobre el tipo de especie del ave estudiada. Nos regala una breve descripción del tipo de ave una vez detectado el tipo de ave.

CONCLUSIÓN

En conclusión, se puede afirmar con certeza que el análisis de las especies en consideración fue un éxito. Como dato complementario, se emplearon datos e imágenes auténticas correspondientes a cada especie estudiada, obtenidas de un repositorio especializado. A continuación, se presentan algunas imágenes representativas del conjunto de datos utilizado para el entrenamiento:

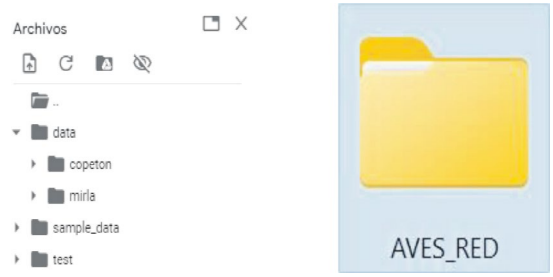


Imagen 19.
Estructura de directorios. Fuente Propia.

Imagen 20.
Directorio de las Aves implicadas en el estudio. Fuente Propia.

En la imagen 19, se distingue claramente el repositorio de datos alojado en Google Colab, mientras que en la otra se muestra el directorio de almacenamiento de los datos.

En la imagen 20 se logran ver las tres carpetas mencionadas anteriormente:

Las carpetas 'Copeton' y 'Mirla' representan las categorías de aves examinadas en este análisis, cada una conteniendo 20 imágenes correspondientes a dichas especies. Por otro lado, la carpeta 'test' sirve como destino para cargar imágenes sobre las cuales se desee realizar predicciones.

REFERENCIAS BIBLIOGRÁFICAS

[1] *WWF,* «¿Por qué Colombia es el país de las aves?,» WWF, 23 mayo 2022. [En línea]. Available: https://www.wwf.org.co/?376931/Porque-Colombia-es-el-pais-de-las-aves.

[2] *eBird,* «Audubon and Cornell Lab of Ornithology,» 2023.

[3] *L. F. N. B. R. L. M. S. P. U. X. H. Echaiz,* El Aporte de la Inteligencia Artificial y las TIC Avanzadas a las Sociedades del Conocimiento, UNESCO Publishing, 2021.

[4] *NaturaLista Colombia,* «Aves del Jardín Botánico de Bogotá,» NaturaLista Colombia, [En línea]. Available: https://colombia.inaturalist.org/guides/6680?view=card.

[5] *L. M. J. G. L. P. C. A. G. A. Óscar Reinoso García,* Ejemplos prácticos de redes neuronales mediante MATLAB y PYTHON, Universitas Miguel Hernández, 2022.

[6] *J. A. L. Sotelo,* Deep Learning: teoría y aplicaciones, Marcombo, 2023.

[7] *J. G. Martínez,* «Reconocimiento visual de aves con Deep Learning,» p. 79, 2024.

[8] *N. Casado Beinat,* «Redes neuronales convolucionales y aplicaciones,» 2022.

[9] *M. S. Patricia Zurita,* «Estado de Conservación de las Aves del Mundo,» Bird Life International, 2022. [En línea]. Available: https://www.birdlife.org/wp-content/uploads/2022/09/SOWB2022_ES_compressed.pdf.

ISSN 2322-8415 / Año 2024 / No. 24 - Vol. 14 / Páginas 38-52 Universidad Libre - Facultad de Ingeniería