Modelo de una red neuronal convolucional para la clasificación de peces asociados a sistemas hídricos del Bagre – Antioquia.

Corporación Universitaria Remington.
Facultad de Ingenierías.
Especialización en Analítica de Datos.

Juan Andrés Burgos Murillo
Asesor: Juan Pablo Vélez
Proyecto de grado.
2025.

Dedicatoria

A mi Abuela (P.S.G.A).

A mí fuerza de voluntad.

A Zoet, quien siempre desde la distancia me espera en casa.

Agradecimientos

A mí esposa Diana y mi Madre Carmen.

A la Empresa Mineros S.A.S.

A la Universidad CES.

Al profesor Juan Pablo Velez.

TABLA DE CONTENIDO

Lista de figuras

Resumen

La implementación de modelos de Deep learning son una de las áreas más exploradas en esta era tecnológica. Dentro encontramos las redes neuronales convolucionales CNN, las cuales permiten a partir de imágenes crear modelos con alto potencial para ser aplicados en diferentes sectores. Por ende, el objetivo principal de este proyecto fue diseñar un modelo de una red neuronal convolucional para la clasificación de peces. Para la creación de este modelo se usó como base MoBileNetV2 y una arquitectura neuronal por capas de extracción de características y clasificación. Para el entrenamiento de la red, se usó una base de datos de 4219 imágenes distribuidas en 10 especies de peces capturados durante el año 2025. El modelo fue evaluado, obteniendo un 87% de precisión, logrando clasificar con alta precisión las especies.

Palabras clave

Peces, Redes Neuronales Convolucionales, MobileNetV2, Reconocimiento de patrones, Deep Learning

1. INTRODUCCIÓN

Lo peces son organismos con características y funciones importantes para la dinámica de los sistemas hídricos, siendo considerados indicadores de la calidad del agua (Ibarra, 2005). Debido a que sus poblaciones pueden cambiar de acuerdo con las condiciones físicas y químicas del agua. Por ende, los monitoreos y la clasificación de los peces asociados a cuerpos de agua son cruciales para la conservación y la toma de decisiones desde el área ambiental.

Tradicionalmente, la identificación de especies se realiza basado en caracteres morfológicos, con métodos en donde se extrae al pez y es enviado al laboratorio para ser identificado, requiriendo de expertos y laboratorios certificados. Sin embargo, aprovechando los avances tecnológicos puede explorarse alternativas que eviten su extracción del hábitat, teniendo así un mejor cuidado en el bienestar de los peces cuando se realicen monitoreos o seguimientos a sus estados poblacionales.

Las redes neuronales convolucionales (CNN), son utilizadas en tareas de clasificación de imágenes en diferentes campos en donde el sector de las ciencias biológicas, específicamente la fauna íctica no esté siendo explorada con métodos o técnicas de Deep Learning. Por lo cual es importante que se haga uso de estas técnicas y aprovechando la aplicabilidad que pueden tener las redes neuronales convolucionales, entrenar modelos

con la capacidad de clasificar un pez a partir de un conjunto de imágenes etiquetadas por el componente ictiológico (Biólogo-Ictiólogo, Auxiliares y pescadores locales) apoyándose de la literatura de referencia para que las etiquetas sean precisas.

Aplicar modelos de Deep Learning a este sector de las ciencias biológicas puede convertirse en una oportunidad para ayudar en la toma de decisiones, mejor uso de la información que se revisa y a la planeación de estrategias para la conservación en menos tiempo.

2. MARCO TEÓRICO

2.1 Peces

Los peces son vertebrados acuáticos, que utilizan branquias para obtener oxígeno del agua y poseen aletas con un número variable de elementos esqueléticos llamados radios (Thurman y Webber, 1984). Poseen espina dorsal larga articulada y continua que recorre su cuerpo teniendo un extremo de ella en el cráneo y el otro formando la cola. Las costillas se encuentran articuladas a la espina dorsal junto con los huesos de las aletas completando así el esqueleto, (Rodríguez & González, 1984).

Estas especies, son uno de los principales grupos de organismos que viven en los ambientes acuáticos de todo el planeta y son componentes fundamentales para el adecuado funcionamiento y regulación de dichos ecosistemas (Arthington et al. 2010, Holmlund y Hammer 1999, 2004).

2.2 Redes Neuronales Artificiales

Una red neuronal es básicamente una estructura de procesamiento paralelo y distribuido de información, la cual consta de elementos conectados unidireccionalmente por canales llamados conexiones de salida con diferentes ramas que transmiten señales y donde toda la información que se procese dependerá de los valores de entrada (Moreno et al., 2019).

2.3 Redes Neuronales Convolucionales

Es un tipo de red multicapa que consta de diversas capas convolucionales y de pooling (submuestreo) alternadas, y al final tiene una serie de capas full-connected como una red perceptron multicapa. La entrada de una red capa convolucional suele ser, generalmente, una imagen m x m x r, donde m es tanto la altura como el ancho de la imagen y r es el número de canales. El aprendizaje es ponderado, los pesos son sumados con la finalidad de proporcionar o generar una activación de nodo, la cuales denominada unidad lógica de umbral. La salida es relativa, las capas de neuronas entregan información de una capa a otra, la eficiencia del aprendizaje generalmente disminuye a medida en que se profundiza el aprendizaje (Anderson, 2007).

Este tipo de red es uno de los algoritmos más populares para Deep Learning, es un tipo de aprendizaje automático en el que un modelo aprende a realizar tareas de clasificación directamente a partir de imágenes, videos, textos o sonidos, siendo especialmente útiles para localizar patrones en imágenes con el objetivo de reconocer objetos, caras o escenas. (Naar & Barreto, 2019).

2.4 MobileNetV2

Es una arquitectura (Figura 2-1) de red neuronal diseñada por Google que logra mejorar la eficiencia computacional sin sacrificar precisión. Puede usarse como extractor de características, tomando una representación comprimida, la expande, la filtra con convoluciones separables en profundidad y la proyecta nuevamente a baja dimensión con una convolución lineal.

```
keras.applications.MobileNetV2(
    input_shape=None,
    alpha=1.0,
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
    name=None,
)
```

Figura 2-1. Instancias de la arquitectura de MobileNetV2
Fuente: https://keras.io/api/applications/mobilenet/

3. PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

La ictiología, como rama de la biología encargada del estudio de los peces, tiene poca información disponible asociada a la aplicabilidad de modelos de Deep learning. Aun cuando se conoce que la implementación de técnicas clásicas para la diferenciación de especies puede llevar a casos en donde es necesario extraer al animal de su ambiente y sacrificarlo, esto aumenta costos de operación para identificar algunas especies debido a sus similitudes en los congéneres. Los avances en inteligencia artificial están siendo enfocados a soluciones empresariales y científicas en donde el sector de la ictiología se está quedando con pocas aplicaciones en el ámbito del aprendizaje automático.

A pesar de los avances en las ciencias de la computación, no existen modelos entrenados para la clasificación de los peces que se encuentran asociados a los sistemas hídricos de los Caños el Pital y el Guamo, ubicados en el Bagre – Antioquia.

El constante crecimiento de herramientas tecnológicas y su aplicación en diferentes campos resulta muy útil e importante para lograr optimizar recursos y tiempo. Dentro del campo de las ciencias biológicas aún existen muchas áreas que deben ser exploradas, buscando estrategias para aplicar técnicas de Deep Learning. La ictiología una rama de la biología encargada del estudio de los peces y su interacción con el ambiente, son objeto de seguimiento y para entender la dinámica de estas especies en el ambiente se suelen aplicar

clasificaciones taxonómicas en base a características morfológicas, ahora bien lo anterior es una oportunidad para que se apliquen técnicas de clasificación basadas en aprendizajes profundos, buscando que redes neuronales aprendan a diferenciar las diferentes especies presentes en un área delimitada en base a datos etiquetados por un grupo de ictiólogos. Al aplicar estas tecnologías a la clasificación y agrupación de especies, se puede aprovechar su potencial para mejorar la precisión y la eficiencia de la clasificación de especies.

Con este proyecto se creó un dataset de imágenes de la fauna íctica, convirtiéndose en un recurso digital pionero para próximas investigaciones y aplicaciones. A demás la aplicación de modelos de Deep Learning en el ámbito de las ciencias biológicas permite incluir esta área, llevándolo a la vanguardia en los procesos de clasificación taxonómica.

4. OBJETIVOS

4.1 Objetivo General

- Diseñar un modelo de red neuronal convolucional para la clasificación de peces asociados a sistemas hídricos del Bagre – Antioquia.

4.2 Objetivos Específicos

- Construir un conjunto de datos etiquetados con las imágenes de los peces que han sido capturados en los sistemas hídricos monitoreados.
- Dividir el conjunto de imágenes para el entrenamiento con un 80% y 20% para validación.
- Definir la arquitectura que se ajuste al conjunto de datos obtenidos.
- Evaluar la red neuronal con métricas de validación como el F1 score y la matriz de confusión.

5. METODOLOGIA

Para la ejecución de este trabajo de grado se utilizarán los registros fotográficos de la fauna íctica que fue capturada durante el año 2025 por el grupo de ictiología que realiza labores de rescate y monitoreo bajo los lineamientos de la Universidad CES para la empresa mineros.

5.1 Recursos informáticos

Para la elaboración de la red neuronal se utilizó el lenguaje de programación Python 3.12.12, con los frameworks Tensorflow 2.18, junto a los módulos y sus librerías estadísticas. Mientras que, para la construcción de gráficas se utilizó Matplotlib.pylot junto a Seaborn (Figura 5-1). Para una mejor organización, el modelo fue realizado dentro de Jupyter Notebook.

```
import tensorflow as tf
from keras.src.legacy.preprocessing.image import ImageDataGenerator
from keras.applications import MobileNetV2
from keras import layers, models
from sklearn.metrics import classification_report, confusion_matrix
from keras.models import load_model
from keras.preprocessing import image
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
✓ 0.2s
```

Figura 5-1. Importación de módulos y herramientas necesarias para el modelo.

5.2 Preparación de conjunto de datos

Para que las fotografías pudieran ser utilizadas en el modelo, se revisaron teniendo en cuenta una buena claridad, posición del pez (con orientación hacia la izquierda) y se recortaron aquellas que tenían datos que pudieran generar ruido como, etiquetas dentro de la foto con numeración o letras. Con lo anterior se logró obtener 4291 imágenes distribuidas en 10 especies (10 subcarpetas).

5.2.1 Redimensionamiento y Carga de las imágenes

Debido a la variación en los tamaños de las fotografías, mediante código se redimensionaron todas las fotografías, defiendo un tamaño de 224 x224 pixeles (Figura 5-2). A demás se configura el número de lotes, Batch_Size = 32, que serán las imágenes que el modelo observará antes de conocer los pesos, el directorio local donde se ubicó la base de datos y la normalización de los pixeles en escala 1. /255, previniendo así que el modelo interprete erróneamente y se ajuste a un modelo base MobileNetV2.

```
IMG_SIZE = (224, 224) # se define el tamaño para todas las imagenes
BATCH_SIZE = 32 #numero de imagenes que verá el modelo antes de ver los pesos
DATASET_DIR = "C:/Users/Juan/Tesis/BD_Peces" # directorio del set de datos

#Configuracion del procesamiento de las imagenes

datagen = ImageDataGenerator(
    rescale=1./255,  # aqui normalizamos los pixeles
    rotation_range=25,
    zoom_range=0.25,
    horizontal_flip=True,
    brightness_range=[0.8, 1.2],
    validation_split=0.2
)
```

Figura 5-2. Configuración y carga de las imágenes.

5.3 División del conjunto de datos

Para el entrenamiento del modelo, se realizó una división del conjunto de imágenes (Figura 5-3) en dos subconjuntos:

- 80% del total se asignaron al entrenamiento, permitiendo al modelo aprender los patrones visuales relevantes.
- 20% restantes se reservaron para la fase de pruebas, con el fin de evaluar la capacidad de generalización del modelo sobre datos no vistos durante el entrenamiento.

```
train_generator = datagen.flow_from_directory(
    DATASET_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

val_generator = datagen.flow_from_directory(
    DATASET_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

num_classes = train_generator.num_classes
```

Figura 5-3. División del conjunto de datos para entrenamiento

5.4 Arquitectura

Debido al número de fotos y la importancia de las variaciones morfométricas de los peces, se utilizó como modelo base entrenado MobileNetV2, permitiendo extraer y detectar patrones básicos de estos. En esta primera etapa de la arquitectura, se congelaron las capas finales del modelo base, evitando generar ruido o sobre entrenamiento (Figura 5-4).

```
#Arquitectura el cual tendra una base de mobileNetV2
base_model = MobileNetV2(
    input_shape=(*IMG_SIZE, 3),
    include_top=False, # aqui se excluye las capas finales de este modelo ya que no es objetivo del proyecto
    weights='imagenet'
)

base_model.trainable = False  # capas iniciales se congelan
```

Figura 5-4. Configuración del modelo base MobileNetV2.

El modelo creado es de tipo secuencial (Figura 5-5), incorporando las capacidades de un modelo base pre-entrenado (MobileNetV2) y usando transfer learning. Este modelo extrae las características de las imágenes, pero manteniendo congeladas las capas finales de ImageNet. Le sigue una capa GlobalAveragePooling2D, encargada de reducir la dimensionalidad del mapa de características y transformándolas en un vector. El cual, se pasa a una capa Densa con 256 neuronas y activándose con la función 'relu', para aprender las combinaciones de las características extraídas.

Se incorporó posteriormente una capa Dropout con una tasa de 0.4, con la función de apagar las neuronas (un 40%) con la finalidad de evitar sobreajustes. Finalmente, se añade otra capa Densa con activación softmax, que produce una distribución de probabilidad de las 10 especies etiquetadas.

```
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),  # reduce la dimensionalidad del mapa de características pasan a ser un vector
    layers.Dense(256, activation='relu'), # capa de activación
    layers.Dropout(0.4), # apaga aleatoriamente el 40% de las neuronas durante el entrenamiento evitando sobreajuste
    layers.Dense(num_classes, activation='softmax') # capa final de clasificación
])
```

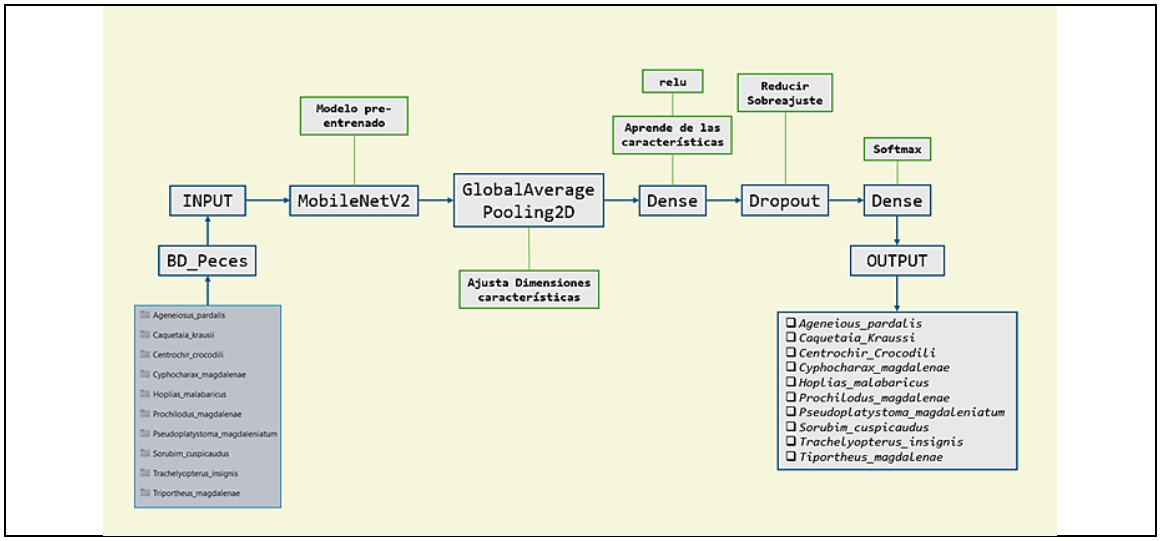


Figura 5-5. Creación del modelo secuencial (A). Diagrama del modelo (B).

Para indicarle a la red neuronal como debía aprender (Figura 5-6), se utilizó el método model.compile() usando el optimizador Adam y las medidas que evaluaron fueron la precisión (accuracy). Para ajustar los pesos internos se utilizó la función (categorical crossentropy).

```
model.compile(
    optimizer='adam',  # optimizador de la clasificación
    loss='categorical_crossentropy', #función de perdida
    metrics=['accuracy'] #metrica de evaluacion
)

model.summary()

 1.3s
```

Figura 5-6. Optimización del modelo con ADAM

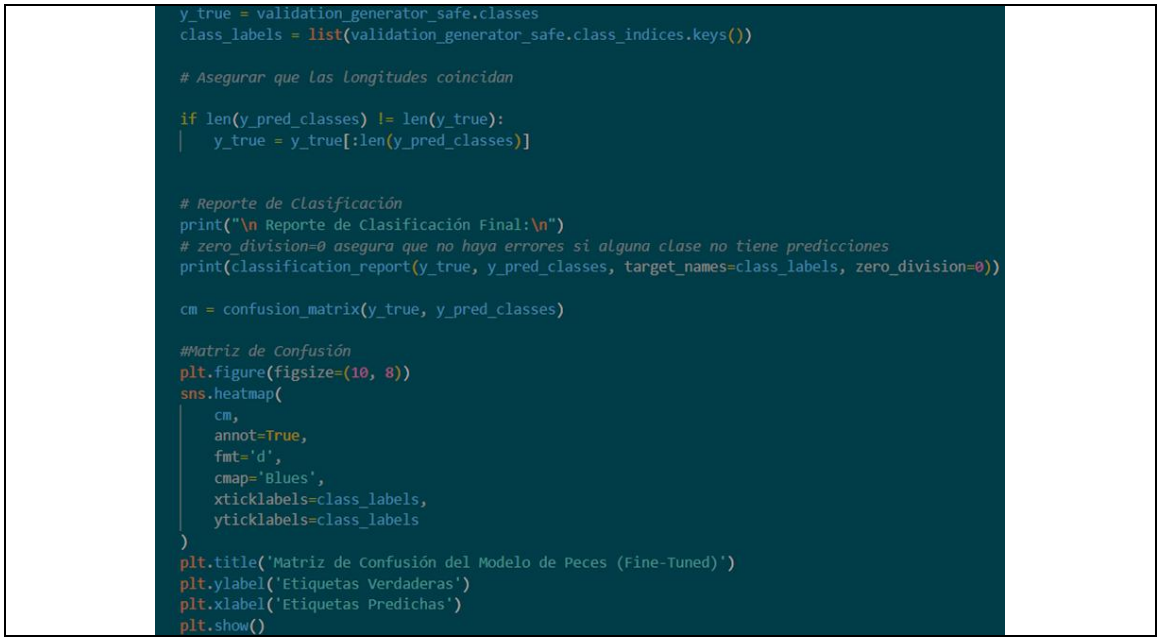
5.4.1 Entrenamiento del Modelo en dos Fases

Con la división del conjunto de datos y la arquitectura del modelo se procedió a su entrenamiento. Para este caso, la red neuronal se entrenó durante 12 épocas o ciclos en la primera fase (Figura 5-7-A). Las capas congeladas de MobileNetV2, ayudaron a mejorar la adaptación del modelo al conjunto de datos. Luego se buscó evitar un sobreajuste en la segunda fase aplicando 'Fine_tunnig' (Figura 5-7-B). Para lo anterior, se descongelaron las ultimas 50 capas del modelo base actualizando los pesos gradualmente, aquí la tasa de aprendizaje se redujo (Learning_rate=1e-5) buscando ser más preciso e identificar características específicas de los peces en las imágenes.

Figura 5-7. Entrenamiento del modelo en dos fases, inicial (A) y final (B).

5.5 Evaluación del Modelo Secuencial

Se definieron las métricas del modelo y se creó un reporte de clasificación (Figura 5-8), además de forma gráfica se creó la matriz de confusión teniendo así una mejor visualización de las posibles inconsistencias.



```
y_true = validation_generator_safe.classes
class_labels = list(validation_generator_safe.class_indices.keys())

# Asegurar que las longitudes coincidan

if len(y_pred_classes) != len(y_true):
    y_true = y_true[:len(y_pred_classes)]


# Reporte de Clasificación
print("\n Reporte de Clasificación Final:\n")
# zero_division=0 asegura que no haya errores si alguna clase no tiene predicciones
print(classification_report(y_true, y_pred_classes, target_names=class_labels, zero_division=0))

cm = confusion_matrix(y_true, y_pred_classes)

#Matriz de Confusión
plt.figure(figsize=(10, 8))
sns.heatmap(
    cm,
    annot=True,
    fmt='d',
    cmap='Blues',
    xticklabels=class_labels,
    yticklabels=class_labels
)
plt.title('Matriz de Confusión del Modelo de Peces (Fine-Tuned)')
plt.ylabel('Etiquetas Verdaderas')
plt.xlabel('Etiquetas Predichas')
plt.show()
```

Figura 5-8 .Código para las métricas de evaluación (reporte) y creación de la matriz de confusión.

6. RESULTADOS Y DISCUSIÓN

6.1 Preparación y División de los datos

El conjunto de datos (imágenes) se configuraron y se ajustaron según a las dimensiones especificadas, dividiendo los lotes de datos para entrenamiento con un total de 3436 fotos, correspondiendo al 80% para entrenamiento del modelo y 855 imágenes se destinaron para las validaciones.

Por otro lado, en estas etapas se implementó la clase ImageDataGenerator de Keras, esto contribuye a un modelo enriquecido por las características generales de las imágenes, mejorando aspecto como la orientación, el brillo, la normalización y escalado, estas últimas fueron esenciales para que lo valores en los rangos definidos fueran estables durante el entrenamiento.

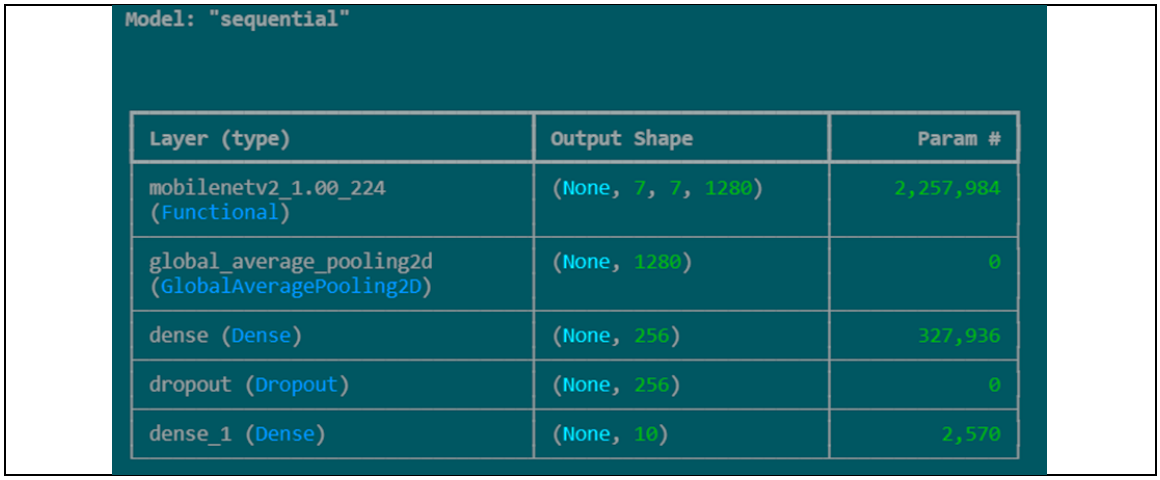
6.2 Arquitectura

La arquitectura implementada muestra que el modelo secuencial (Figura 6-1), en la capa base MobileNetV2 generó mapas de activación de tamaño 7x7 pixeles con 1280 canales o características que la red observa en cada región y contiene 2,257,984 parámetros no entrenables, manteniéndolos congelados durante el entrenamiento. Posteriormente la capa GlobalAveragePooling2D, logró reducir los mapas de activación a un vector

unidimensional de 1280 características iniciando así una transición a la primera capa Dense.

La capa Dense (256), contiene 256 neuronas cada una conectada a las 1280 entradas. Esto permitió que la red aprendiera sobre las combinaciones específicas de las características morfológicas de los peces, con un subtotal de parámetros de 327680 pesos entrenados y un sesgo por cada neurona, lo cual produce en total 327936 parámetros.

La regularización de la capa Dropout, aplicó una tasa del 0.4, en donde esta parte de las neuronas se desactivaron. Finalmente, en la capa de salida Dense es quien clasifica finalmente las especies aprovechando la activación Softmax para que cada imagen de entrada se convierta en 10 probabilidades, es decir las 10 especies clasificadas. Para lo anterior se usan 2570 parámetros entrenados con las probabilidades a cada espécimen.



Model: "sequential"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 256)	327,936
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2,570

Figura 6-1. Resumen de los parámetros en la arquitectura del modelo.

6.3 Entrenamiento de la Red Neuronal

6.3.1 Primera Fase

En esta fase del entrenamiento se observó (Figura 6-2), que la red neuronal en la primera época tiene una precisión (accuracy) del 0.6604 y termina en 0.9715, es decir pasa del 66% al 97% indicando que el modelo está aprendiendo considerablemente sobre las características del conjunto de imágenes de entrenamiento. La pérdida (loss) del entrenamiento va de 1.062 hasta 0.087, reflejando una buena tendencia de aprendizaje. La precisión de la validación (val_accuracy) comienza con un 74% y finaliza en el último ciclo con 85%, es decir el modelo está aprendiendo considerablemente. Mientras que la perdida de validación (val_loss), presentó más variación indicando levemente sobreajuste, sin embargo, en las últimas dos épocas se estabiliza.

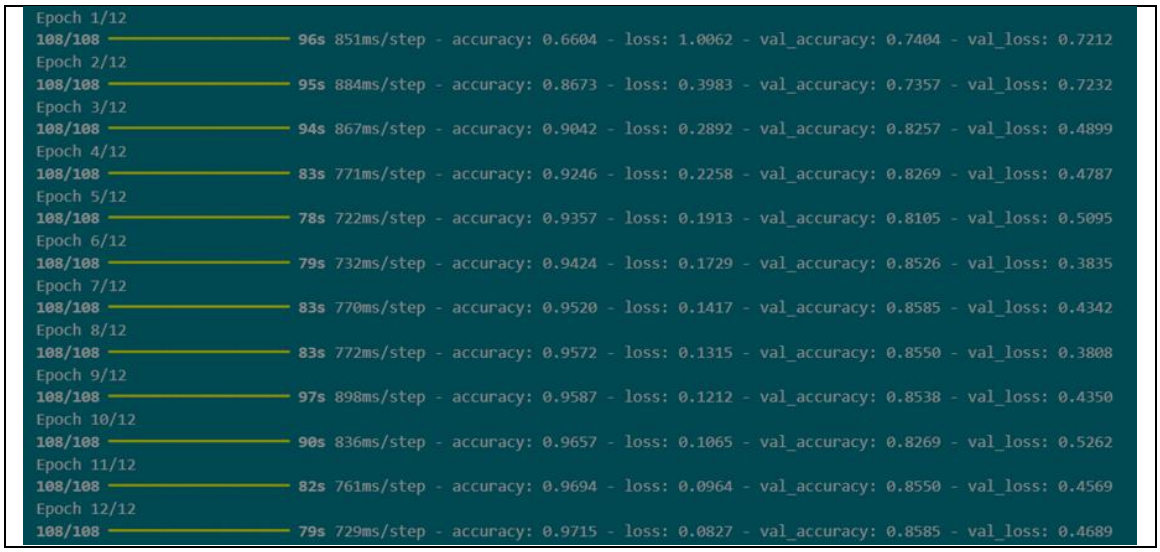


Figura 6-2. Entrenamiento en fase inicial.

El comportamiento de la precisión y su validación (Figura 6-3-A), mostraron un alto sobreajuste reflejado por la separación al final de las líneas por más del 10%. Este comportamiento fue similar para la perdida y su validación (Figura 6-3-B). Este resultado, nos sugiere que es necesario realizar un leve ajuste al entrenamiento.

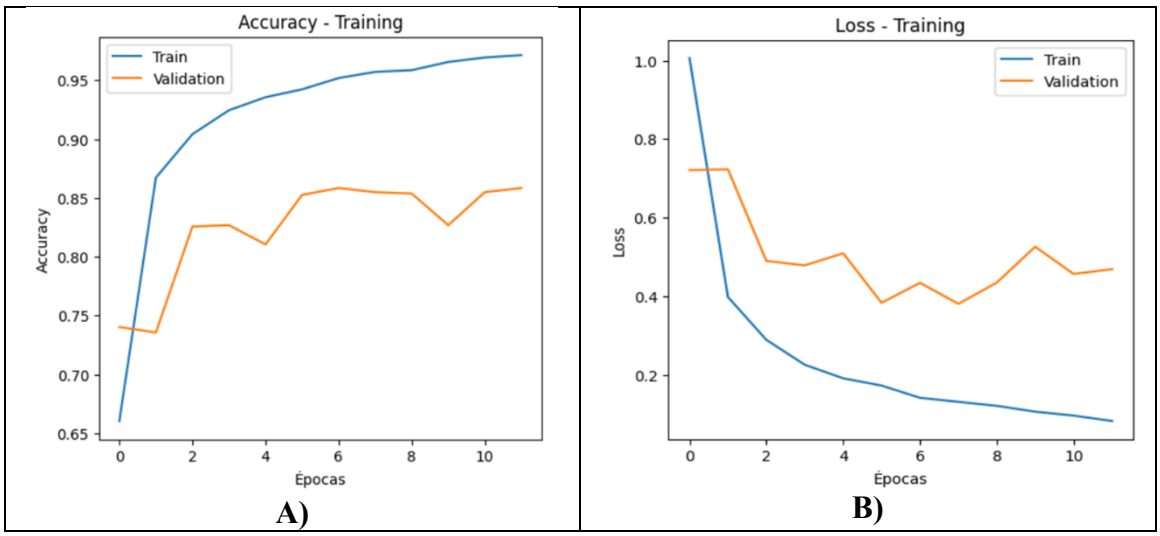


Figura 6-3. Comportamiento del modelo en la fase inicial de entrenamiento. Precisión (A) y Perdida (B).

6.3.2 Segunda Fase

Debido al entrenamiento previo se observa que en esta fase (Figura 6-4), el modelo empieza con una precisión (accuracy) mucho mejor 0.8405 (84%) y finaliza en 0.9854 (98%), mientras que la validación de la precisión comienza en 0.8175 (81%) y termina en 0.9041 (90%). Mientras que la perdida (loss) comenzó en la primera época con 0.5071 y finaliza en un valor inferior igual a 0.0530. La validación de la perdida fue de 0.6514 y termino en un mínimo de 0.3423. lo anterior muestra que, el modelo ha mejorado en su precisión y reduciendo su perdida, lo cual puede reflejarse en predicciones más confiables y menos dispersas.

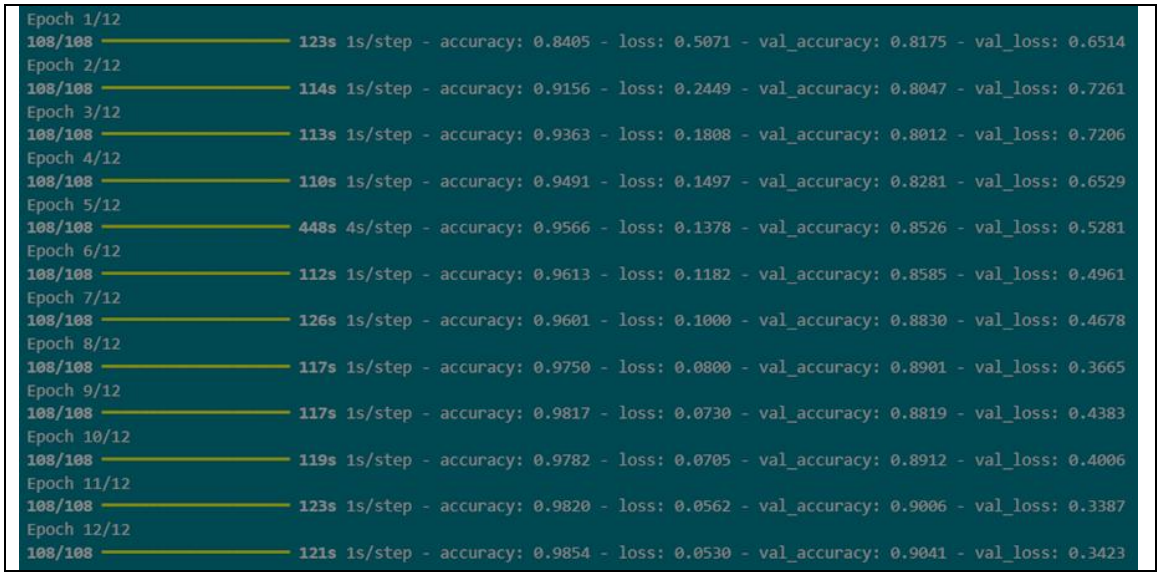


Figura 6-4. Entrenamiento en fine_tunning.

El comportamiento de esta segunda fase refleja que la precisión y su validación (Figura 6-5-A), se comportan paralelamente en lo ciclos mejorando la capacidad de predecir del modelo con un 90% y mejorando en 5 puntos aproximadamente con respecto a la primera fase de entrenamiento. Este comportamiento se refleja similarmente en la perdida (Figura 6-5-B), la cual baja gradualmente, aunque se presentó una variación entre la época 6 a la 8, pero logra estabilizarse. Se puede inferir que, el fine-tuning fue altamente efectivo, mejorando la precisión y reduciendo la pérdida en la validación, lo que indica una mejor capacidad de predecir imágenes que no ha visto.

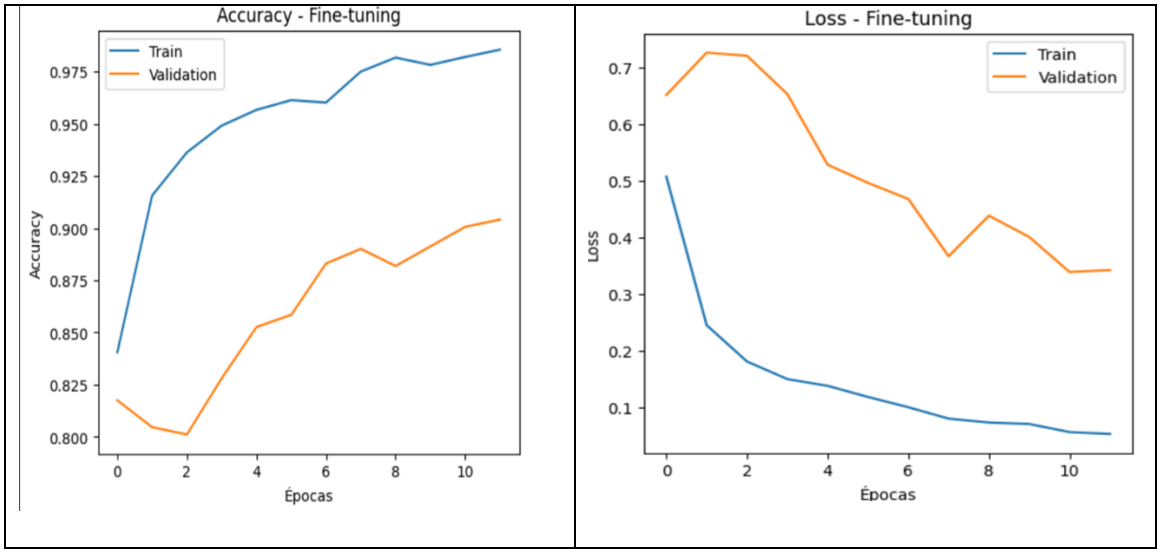


Figura 6-5. Comportamiento del modelo en la segunda fase de entrenamiento. Precisión (A) y Perdida(B).

7. EVALUACIÓN DEL MODELO

Las métricas de evaluación para las 10 especies (Figura 7-1), evidencian que el modelo tiene una precisión del 87%, es decir que la red neuronal convolucional diseñada puede realizar predicciones con buena precisión.

Las especies con mayor precisión para el modelo son *Caquetaia krausii*, *Pseudoplatystoma magdaleniatum* y *Prochilodus magdalenae*, alcanzando altos valores de F1-Score 0.97, 0.92 y 0.90, respectivamente. Esto refleja que el modelo d la red neuronal, tiene la capacidad de reconocerlas con alta exactitud y confinza. Por otro lado, se identificaron algunas especies con mayor nivel de confusión, particularmente *Cyphocharax magdalenae* y *Hoplias malabaricus*, que presentaron bajos F1-Scores de 0.75 y 0.79, respectivamente.

	precision	recall	f1-score	support
Ageneiosus_pardalis	0.95	0.92	0.93	85
Caquetaia_krausii	1.00	0.95	0.97	91
Centrochir_crocodili	1.00	0.80	0.89	79
Cyphocharax_magdalenae	0.64	0.90	0.75	83
Hoplias_malabaricus	0.89	0.71	0.79	77
Prochilodus_magdalenae	0.92	0.88	0.90	96
Pseudoplatystoma_magdaleniatum	0.92	0.91	0.92	92
Sorubim_cuspicaudus	0.85	0.86	0.86	81
Trachelyopterus_insignis	0.80	0.84	0.82	92
Triportheus_magdalenae	0.86	0.92	0.89	79
accuracy			0.87	855
macro avg	0.88	0.87	0.87	855
weighted avg	0.89	0.87	0.87	855

Figura 7-1. Reporte de la clasificación del modelo y métricas.

Al testear el modelo con imágenes no vistas, se comprueba su alta precisión (Figura 7-2), clasificando correctamente especies de gran interés para la conservación y el cuidado, además de que algunas de las especies entrenadas para el modelo también son objeto de interés para su protección debido a la disminución de sus poblaciones naturales como los son el bocachico (*Prochilodus magdalenae*), el bagre rayado (*Pseudoplatystoma magdaleniatum*) y el blanquillo (*Sorubim cuspicuadus*).

A)

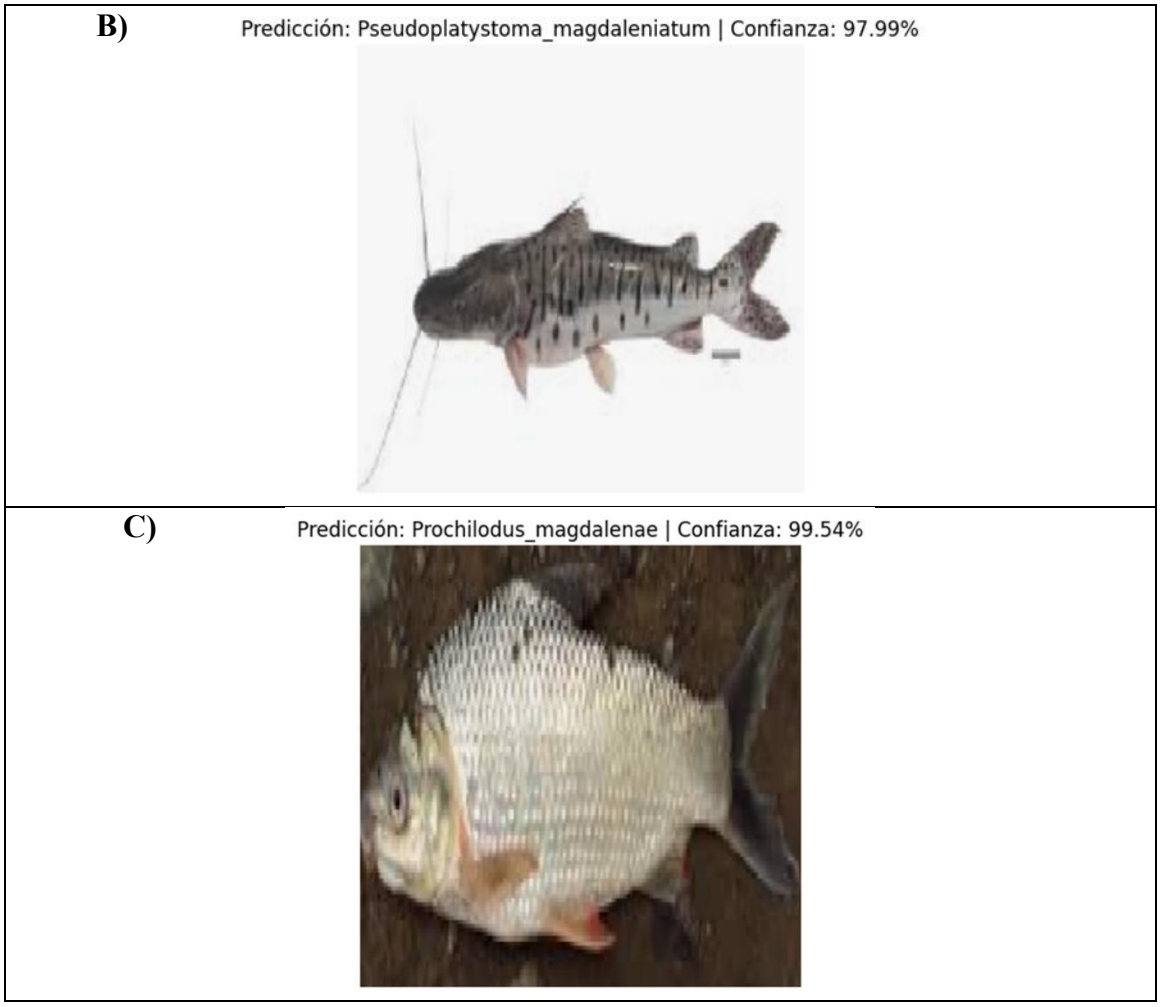


Figura 7-2. Resultado de clasificación correcta para las especies *Sorubim cuspicaudus* (A), *Pseudoplatystoma magdaleniatum* (B) y *Prochilodus magdalenae* (C).

Se creó la matriz de confusión (Figura 7-3) para revisar como se distribuyen las especies en el modelo y cueles podría estar confundiendo. La matriz muestra en color azul fuerte las especies con mejor capacidad de clasificarlas correctamente y en azul más claro aquellas que son confundidas fácilmente.

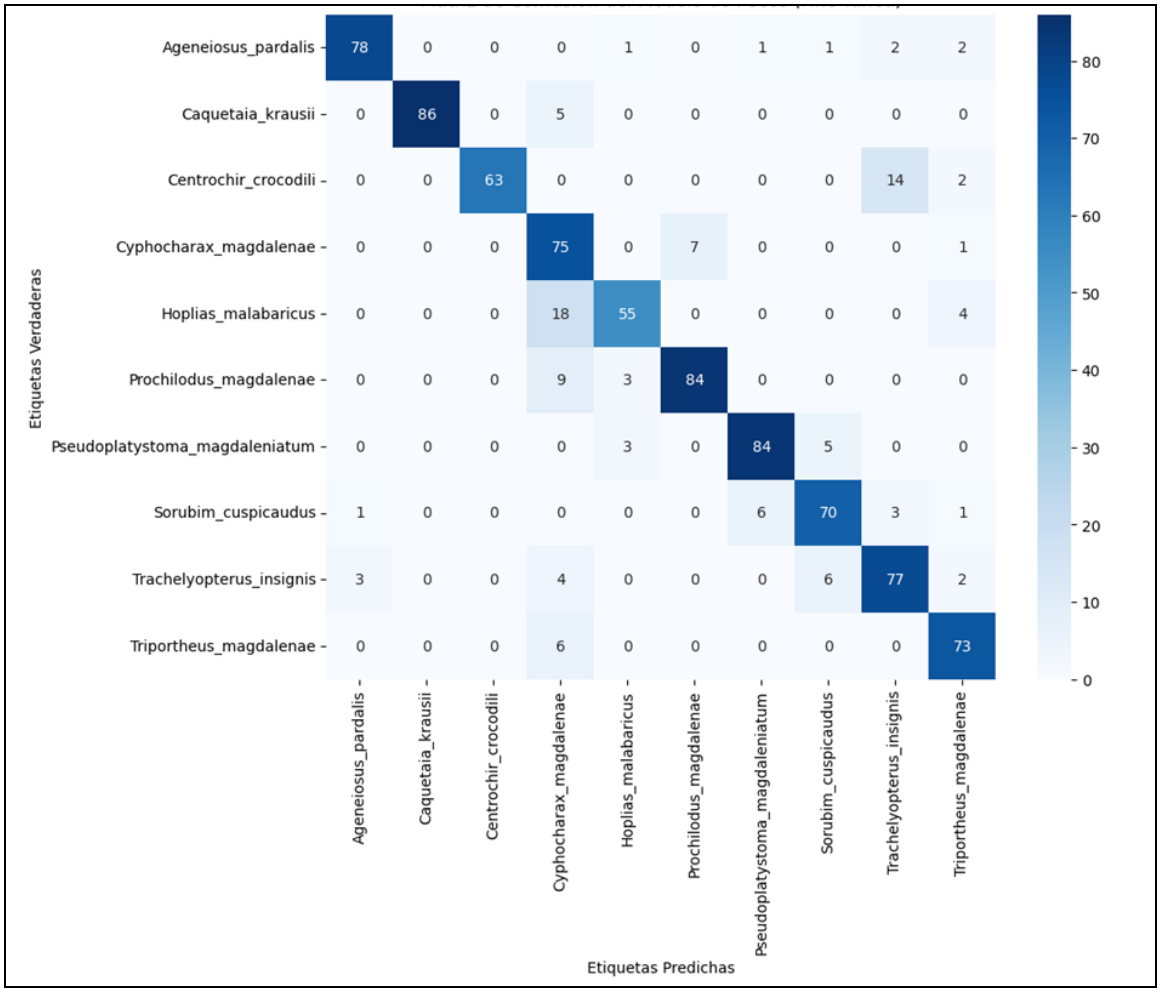


Figura 7-3. Matriz de Confusión.

8. CONCLUSIONES

Se logró crear un set de datos de imágenes que permitieron el entrenamiento de un modelo de red neuronal para la clasificación de 10 especies de peces propuestos. A demás, la arquitectura con un modelo base (MobileNetV2), facilita la extracción de características para estas redes convolucionales con bases de datos con un número no tan alto de imágenes (solo 4291).

La alta precisión de la red neuronal convolucional demuestra que es capaz de ser aplicada en contextos reales, apoyando labores de clasificación de peces. sin embargo, es importante seguir entrenando nuevas versiones de la base de datos con registros de otras especies para que el modelo tenga más capacidad de ayudar a clasificar otras especies de gran interés para la conservación.

La ictiología es una rama de las ciencias biológicas que tiene como reto y una oportunidad muy poco estudiada, para aplicar modelos de Deep Learning que ayuden a la solución o mejora de investigaciones y actividades ambientales enfocadas a la conservación.

9. REFERENCIAS BIBLIOGRÁFICAS

Anderson J. A. (2007). Redes Neuronales. México DF: Alfaomega.

Arthington, Á. H., Naiman, R. J., Mcclain, M. E., & Nilsson, C. (2010). Preserving the biodiversity and ecological services of rivers: new challenges and research opportunities. Freshwater biology, 55(1),116. DOI: 10.1111/j.13652427.2009.02340.

Holmlund CM, Hammer M. 2004. Effects of fish stocking on ecosystem services: An overview and case study using the Stockholm Archipelago. Environmental Management, 33(6): 799–820. DOI:10.1007/s00267-004-0051-8.

Ibarra, A. A. (2005). Los peces como indicadores de la calidad ecológica del agua.

Holmlund CM, Hammer M. (1999). Ecosystem services generated by fish populations. Ecological Economics, 29(2): 253-268. DOI: 10.1016/S0921-8009(99)00015-4.

Moreno, L., Garrido, S., & Copaci, D. (2019). Fundamentos de las Redes Neuronales L6.

Naar Pérez, A., & Barreto Martínez, F. (2019). Modelo de red neuronal convolucional para el diagnóstico de neumonía en imágenes radiológicas (Doctoral dissertation, Universidad del Sinú, seccional Cartagena).

Thurman, H.V. & Webber, H. H. (1984). Marine Biology. Charles E. Merrill Publishing C.A. Bell and Howell Co. Columbus, Ohio.