



GRADO EN INGENIERÍA INFORMÁTICA Y ADE

PROYECTO FINAL - REDES NEURONALES

Análisis y Clasificación de Peces mediante Machine Learning

JACOBO CHAVES - PEDRO LATASA

CUNEF

MAYO 2025

Contenidos

1. Introducción	2
2. Solución Propuesta	2
3. Resultados + Estado del arte	3
4. Bibliografía	5

1. Introducción

Uno de los principales retos de la pesca comercial es la identificación precisa de las especies capturadas. La falta de tiempo o conocimientos puede llevar a capturas accidentales (*bycatch*) y al descarte de especies valiosas o protegidas, con consecuencias económicas, ecológicas y legales.

Esta tecnología podría integrarse en embarcaciones pesqueras mediante cámaras y sistemas de visión artificial, facilitando la clasificación automática a bordo. En este proyecto nos centramos exclusivamente en el desarrollo del modelo software, sin abordar la infraestructura física necesaria para su implementación.

Una herramienta automática de clasificación de peces a partir de imágenes puede reducir estos descartes, y facilitar el cumplimiento de normativas. Este proyecto propone una solución basada en herramientas de Machine Learning, aplicadas al dataset *A Large Scale Fish Dataset*, que contiene 9.000 imágenes clasificadas en 9 especies.



Figura 1: Muestra de imágenes de las diferentes especies de peces del dataset.

2. Solución Propuesta

Los problemas de clasificación de imágenes los hemos abordado con diferentes herramientas. Modelos como ResNet han sido fundamentales. Además de arquitecturas como EfficientNetB1 han ganado protagonismo por su eficiencia computacional y rendimiento.

En estudios relacionados con clasificación de peces, hemos empleado desde CNN simples hasta modelos transferidos y ajustados (“fine-tuned”) pre-entrenados en ImageNet. La aplicación de data augmentation también es común para mejorar la generalización en datasets con variabilidad visual.

Nuestra solución se basa en comparar diferentes modelos:

- **EfficientNetB1**, como modelo principal de clasificación profunda.

Preprocesamiento:

- Reescalado de imágenes a 240x240.
- Normalización de valores RGB.
- Data augmentation (rotaciones, flips, cambios de brillo).

Protocolo experimental:

- 70 % entrenamiento, 15 % validación, 15 % prueba.
- Fine-tuning sobre EfficientNetB1 preentrenado en ImageNet.
- Criterios de evaluación: accuracy, precision, recall, F1-score.

3. Resultados + Estado del arte

Inicialmente entrenamos nuestros modelos en local, pero lo cambiamos por Kaggle por varias ventajas clave: disponibilidad de GPUs más potentes (el entorno de Kaggle permite hasta el doble de tiempo de uso de GPU gratuito), facilidad en la gestión de datos (el dataset ya se encuentra alojado en la plataforma) y la mayor comodidad a la hora de importar imágenes y gestionar notebooks.

También desarrollamos una red propia (CNN simple), pero a medida que explorábamos distintas estrategias para mejorar la eficiencia del modelo, consultamos proyectos similares en Kaggle. Tras analizar distintas alternativas, concluimos que utilizar la arquitectura EfficientNetB1 era la opción más adecuada para nuestro caso.

Durante el proceso de entrenamiento, comenzamos con la red propia, con la cual obtuvimos una precisión cercana al 95 %. Posteriormente, al implementar la arquitectura EfficientNetB1 y ajustar diversos hiperparámetros, alcanzamos una precisión del 100 % en la validación. Entre los ajustes realizados destacan la reducción del *learning rate* de 0.001 a 0.0005 (ya que al ser tan alto el ”loss” aumentaba) y una disminución progresiva del número

de épocas, tras observar que un menor número ofrecía mejores resultados y evitaba el sobreajuste. También incorporamos un mecanismo de *early stopping* con *patience* de 5 épocas, permitiendo detener el entrenamiento cuando la métrica de validación dejaba de mejorar, optimizando así el tiempo de cómputo.

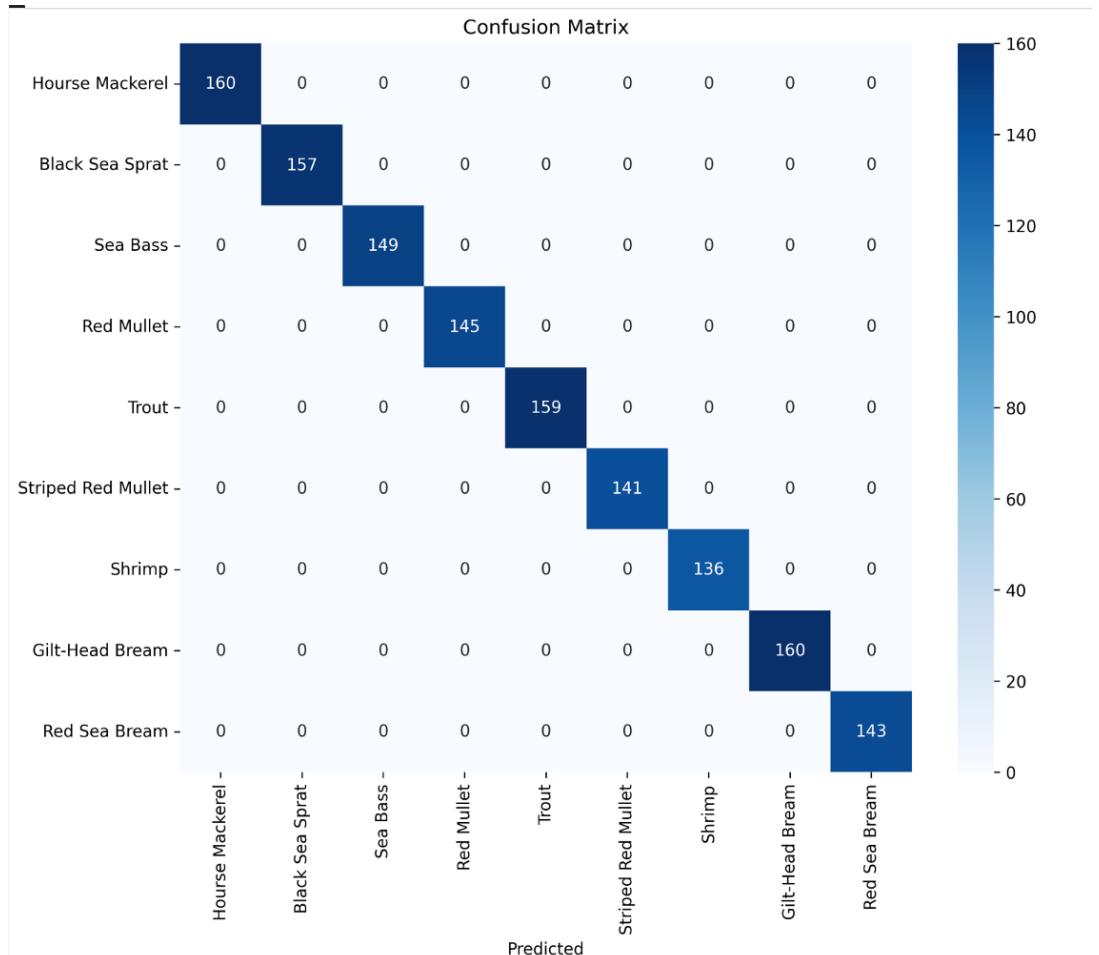


Figura 2: Matriz de confusión del modelo EfficientNetB1

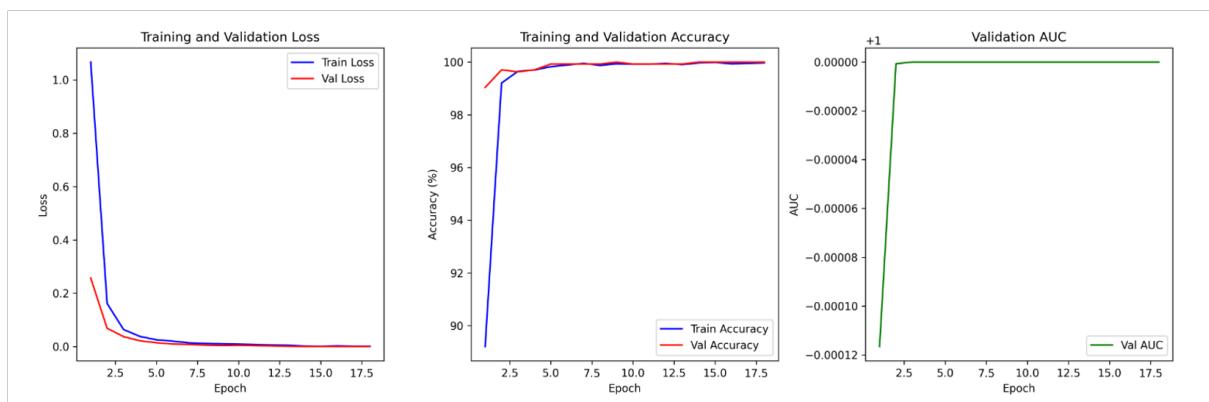


Figura 3: Training/Validation Loss, Accuracy y AUC

4. Bibliografía

Referencias

- [1] Scikit-learn. *Machine Learning in Python*. Disponible en: <https://scikit-learn.org>
- [2] Pandas. *Data analysis and manipulation tool*. Disponible en: <https://pandas.pydata.org>
- [3] Matplotlib. *Visualization library in Python*. Disponible en: <https://matplotlib.org>
- [4] NumPy. *Numerical computing tools*. Disponible en: <https://numpy.org>
- [5] Tan, M. y Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946*.
- [6] Pedro Latasa y Jacobo Chaves. *CNN Fish Classification - Kaggle Notebook*. Disponible en: <https://www.kaggle.com/code/pedromaralatasa/cnn-fish-classification>
- [7] Pedro Latasa y Jacobo Chaves. *CNN Fish Classification - GitHub Repository*. Disponible en: <https://github.com/PedroLatasa/cnn-fish-classification.git>