Rare Species Image Classification

Deep Learning Course Project Report

Group Identification

Group 12

- Yan Sidoryk - 20222004
- Henry Lewis - 20222002
- Abdul Rehman Khan - 20231738
- Lowie De Wever - 20231733

1. Introduction

This project focuses on the challenging task of classifying rare species images into 202 taxonomic families across 5 phyla. The dataset, comprising of approximately 12,000 images sourced from the Encyclopedia of Life (EOL), presents several difficulties, including severe class imbalance, the presence of substantial noisy data (e.g., sketches, X-rays), and high inter-class visual similarity among closely related species.

2. Data Exploration

The original dataset of 11,983 images exhibited extreme class imbalance across the 5 phyla and 202 families. Phylum Chordata was highly dominant (~83%), while Echinodermata was severely underrepresented (~0.5%). At the family level, the imbalance was clear, with the largest class (Cercopithecidae) having approximately a 10:1 ratio of images compared to the smallest.

Initial analysis of image dimensions led to the selection of a 500x375 pixel resolution, balancing detail and computational load, given the clustered 4:3 aspect ratio of most images. We also identified and removed unsuitable data types: 183 greyscale images (mostly sketches/diagrams) and 178 CMYK images (mostly primate skull X-rays). Only two valid CMYK photographs (a turtle and parrots) were retained and converted to RGB.

3. Preprocessing and Outlier Detection

Given the presence of numerous problematic images (signs, text documents, maps, food, humans without animals, among others), we systematically evaluated multiple outlier detection approaches to help deal with these.

3.1 Outlier Detection Methods

YOLO Person Detection: We hypothesised that images containing humans without animals could be identified using YOLOv8 person detection [4]. However, at confidence threshold 0.85 the detector primarily flagged images of researchers holding animals, which are images we wished to retain. Alternative prompts and thresholds failed to yield useful filtering criteria, causing us to look for alternative methods.

ImageNet Classification: Using InceptionV3 [5] pretrained on ImageNet, we attempted to identify non-animal images. This approach flagged 4,675 images (~40%) as non-animals, far too aggressive given that many valid wildlife photographs were misclassified as outliers due to domain differences between ImageNet and our rare species imagery.

CLIP Semantic Scoring: We employed OpenAI's CLIP model [2] to compute semantic similarity between images with two prompt categories: "good" prompts (wildlife/animal photographs) and "bad" prompts (x-rays, diagrams, text, food, humans). Images where the "bad" similarity exceeded the "good" similarity (semantic quality < 0) were flagged as outliers. This approach identified 640 outliers with high precision, successfully capturing x-rays, text documents, diagrams, and irrelevant content, which led us to drop these images.

Distance To Centroid: After generating embeddings using EfficientNetB4 [1], we computed class centroids from training data and measured each image's Euclidean distance to its class centroid. Images beyond the 95th percentile were removed in a first pass, followed by a second pass at the 97.5th percentile. This method effectively captured obvious outliers missed by CLIP.

FIGURE 1: Samples of outliers removed using the CLIP semantic scoring approach.

FIGURE 2: Samples of outliers identified and removed using the distance to centroid method.

3.2 Final Preprocessing Pipeline

Our final preprocessing pipeline: (1) Remove all greyscale and CMYK images except 2 valid CMYK, (2) Remove CLIP outliers (semantic quality < 0), (3) Split into train/test (85%/15%, ensuring stratification), (4) Compute embeddings and remove distance-to-centroid outliers (>95th, then >97.5th percentile), (5) Split the remaining Training set into Training/Validation (90%/10%, stratified). After preprocessing, 10,140 images remained (from 11,983 original), split as: Training: 7,780 (77%), Validation: 865 (9%), Test: 1,495 (15%).

4. Data Augmentation

We experimented extensively with augmentation strategies including brightness shifts, contrast adjustments, and colour channel modifications. However, visual inspection of augmented batches revealed that aggressive settings produced unrealistic images, overly bright, overly dark, or with unnatural colour casts, that did not reflect the original dataset's characteristics. Since the source images already exhibited reasonable consistency in lighting and colour, we concluded that such augmentations would introduce noise rather than useful variation.

We therefore opted for a more conservative augmentation strategy targeting only geometric transformations: horizontal flip (animals may face either direction), rotation ±15° (camera

angle variation), and zoom ±10% (distance variation), to hopefully capture more of the natural discrepancies seen in the original pictures.

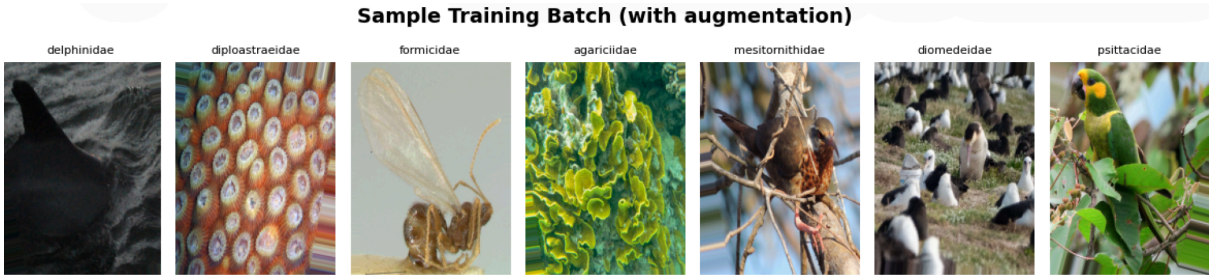

FIGURE 3: Sample training batch with applied conservative augmentation.

5. Model Architecture and Training

5.1 Model Iterations and Comparison

We evaluated several architectural approaches before arriving at our final model. Table 1 summarises the performance of each approach:

Model	Val Accuracy	Notes
Custom CNN (from scratch)	~2.5%	Near random chance
EfficientNetB0 + Dense layers	~63%	Baseline transfer learning
EfficientNetB0 + Tuned head	~79%	After hyperparameter tuning
EfficientNetB4 + Classification only	~83%	No non-linearity (rejected)
EfficientNetB4 + Dense Layer + Tuned head (Final)	~84%	Best approach

Table 1: Model comparison summary

Custom CNN (from scratch): A custom 4-block CNN achieved only ~2.5% validation accuracy after 20 epochs, which is barely above random chance for 202 classes (~0.5%). The lack of meaningful learning confirmed that training a deep CNN from scratch was infeasible given the dataset size (~10,000 images across 202 classes) and necessitated the use of transfer learning.

EfficientNetB0: Our initial transfer learning approach using the smaller EfficientNetB0 [1], which proved to be a lot better of a reference to build upon, reached ~63% validation accuracy. After hyperparameter tuning with Keras Tuner (Hyperband), the optimised architecture achieved ~79% validation accuracy.

EfficientNetB4 with classification layer only: We then tested EfficientNetB4 with only a softmax classification head, achieving an impressive ~83% validation accuracy. However, we rejected this approach because the absence of additional dense layers meant the model contained no trainable non-linearity beyond the frozen pretrained features, effectively performing linear classification on extracted features without learning dataset-specific representations, which limits its generalisation capability.

EfficientNetB4 with tuned head (Final): Adding a trainable dense layer allows the model to learn task-specific feature combinations, leading to improved performance.

5.2 Hyperparameter Tuning

We employed Keras Tuner with Hyperband optimisation to search over: number of dense layers (1-3), hidden units per layer (400-1000), dropout rate (0-0.5), and optimiser (Adam [6], Adagrad [7]). The optimal configuration: single dense layer with 600 units, dropout rate of 0.4, and Adagrad optimiser (learning rate 0.01, preferred for minority class sparsity).

5.3 Final Architecture

EfficientNetB4 [1] (frozen) → GlobalAveragePooling2D → BatchNormalization → Dense(600) → Dropout(0.4) → Dense(202, softmax)

Total parameters: 18,878,193 (1,200,786 trainable)

5.4 Class Weighting

Given severe class imbalance (300 images in the largest class vs. 13 in the smallest), we computed balanced class weights using scikit-learn's compute_class_weight function, ensuring the loss function appropriately penalised errors on minority classes.

5.5 Training Procedure

Main Training (20 epochs): Adagrad optimiser (lr=0.01) with EarlyStopping (patience=3) and ReduceLROnPlateau (factor=0.1, patience=2). Learning rate reduced to 0.001 at epoch 12.

Fine-Tuning (6 epochs): Unfroze the last 5 layers of EfficientNetB4 (excluding BatchNormalization to preserve running statistics) with a very low learning rate (1×10^{-7}) to fine-tune without overwriting the valuable pretrained features. Validation accuracy improved from 81.4% to 81.9%. Early stopping triggered at epoch 6.

6. Experimental Setup

The project leveraged several key libraries: TensorFlow/Keras for model building and training; Keras Tuner (Hyperband) for hyperparameter optimisation; and scikit-learn for tasks like train/test splitting, metric calculation, and class weight generation. For outlier detection, we used Transformers (HuggingFace) to access the CLIP model [2] for semantic scoring and Ultralytics for the YOLOv8 [4] person detection experiments. General data handling relied on PIL, NumPy, and Pandas, while visualisation was handled by Matplotlib, Seaborn, and Plotly.

7. Results

7.1 Model Performance

Metric	Score
Test Accuracy	84.01%
F1 Score (Micro)	84.01%
F1 Score (Macro)	82.88%
F1 Score (Weighted)	83.99%
Test Loss	0.5643

Table 2: Final model results

The close alignment between micro and macro F1 scores indicates reasonably balanced performance across classes despite the imbalance.

7.2 Metadata Enhanced Predictions

As a bonus experiment, we leveraged available taxonomic metadata (phylum) at inference time. Since each phylum contains a known subset of families, we masked logits for impossible family predictions based on the test image's phylum. This constraint improved accuracy from 84.01% to 84.35% and macro F1 from 82.88% to 82.95%.

8. Error Analysis

The confusion matrix showed that most misclassifications occurred between visually similar families within the same phylum, or when multiple subjects are in the image. Errors seem to stem from inherent dataset difficulties rather than a systematic model failure, including images with unusual poses/occlusion, high morphological similarity between families, and cases judged to be visually ambiguous even by human observers. These represent edge cases that would challenge any classifier. Therefore we are very confident in our model and the results yielded.

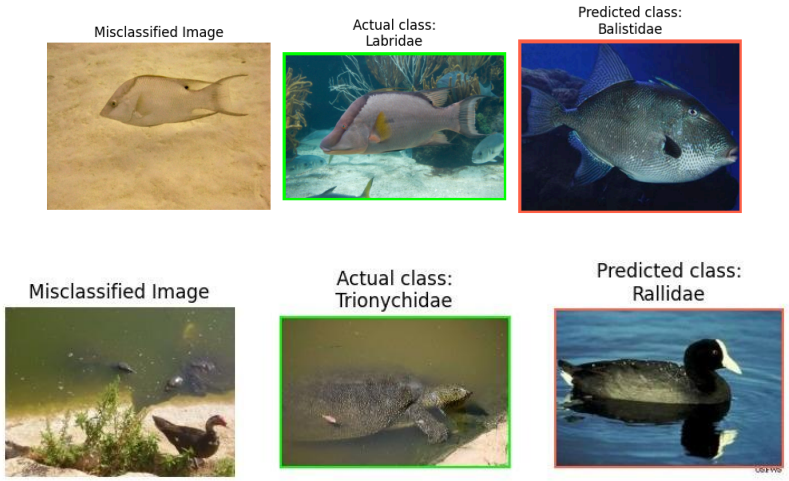

FIGURE 4: Example of Misclassified Images.

9. Future Improvements & Conclusion

Given additional time, we would explore:

- **Class Imbalance Mitigation:** Synthetic data generation (GANs/diffusion models) to augment underrepresented families, or collecting additional images from external wildlife databases.
- **Advanced Outlier Detection:** A multi-stage pipeline combining CLIP [2] (or the domain-specific Bioclip [3]) with anomaly detection models (autoencoders, isolation forests) to catch remaining problematic images.
- **Multi-Subject Handling:** Object detection (YOLO/Faster R-CNN) as preprocessing to isolate individual animals, reducing confusion from multi-species images.

In conclusion, achieving a weighted F1 of 0.84 on a 202-class problem exceeded our expectations and reinforced that success in deep learning depends as much on understanding your data as on choosing the right architecture.

10. References

[1] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," ICML, pp. 6105-6114, 2019.

[2] A. Radford et al., "Learning transferable visual models from natural language supervision," ICML, pp. 8748-8763, 2021.

[3] S. Stevens et al., "Bioclip: A vision foundation model for the tree of life," Proceedings of the IEEE/CVF CVPR, pp. 19412-19424, 2024.

[4] Ultralytics, “YOLOv8: Ultralytics YOLOv8,” 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] C. Szegedy et al., "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9, 2015.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," ICLR, 2015.

[7] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *JMLR*, vol. 12, pp. 2121-2159, 2011.