NOVA IMS Information Management School

MDSAA

Master's Degree Program in

Data Science and Advanced Analytics

Deep Learning

João Santos, 20211691

Nuno Bernardino, 20211546

Rita Serra, 20240515

Rui Lourenço, 20211639

Tiago Pedro, 20240655

Group 6

Contents

1. Introduction

Biodiversity studies depend on the proper classification of biological species. Yet, for rare or underclassified taxa, this can be an arduous undertaking. Thus, we plan to train a deep learning model to classify images of species from a dataset of the Encyclopedia of Life (EOL) collection, BioCLIP, to their rightful family. This is a combination of taxonomical classification and computer vision application in hopes that our contributions could help subsequent researchers in the field with either study or conservation efforts.

The provided dataset will include images with hippopotamus metadata for kingdoms, phylum, family, and rare taxa. The prediction task will be the classification of families with appropriate labels assigned for each classification across the image photos. The challenge comes from extraordinarily rare classes and classes with similar appearances. Thus, we explored three different approaches to image preprocessing, modelling, and hyperparameter tuning. Our ultimate solution implements the most cutting-edge approaches from transfer learning and convolutional neural networks (*CNN's*) for the best-performing, generalizable classifier.

This report describes the process taken to achieve this champion model from dataset creation to assessment, including error analysis and future considerations.

2. Methodology

2.1 Libraries and Other Implementation Details

For this implementation, we used several Python libraries, such as:

- **os** - for file paths;
- **zipfile** - for extracting files in a zip;
- **pathlib** - to manipulate file paths;
- **pandas** - to analyse the tabulated data, we continue.
- **numpy** - for mathematical operations;
- **matplotlib** - to better understand certain conclusions through graphs;
- **seaborn** - to visualise statistical data;
- **PIL** - to load and process images;
- **sklearn** - for dividing data into sets and also used to evaluate model performance;
- **rembg** - to remove the background from images;
- **IPython.display** - to display images directly in the development environment;
- **shutil** - to copy, move and delete files;
- **TensorFlow** and **Keras** - for building, training and tuning models.

Important notes:

If you would like to run the code, please visit the GitHub repository below, as we were unable to upload all the required files, as the weight of the zipped folder exceeded the weight allowed by Turnitin.

https://github.com/TiagoPedrotkd/DL_EOLP

It is important to note that to run our implementation correctly, it is necessary to insert the zipped folder that contains all the animal families and their respective images and also the metadata (*rare_specie.zip*) inside the Raw folder that is currently empty.

2.2 Data Analysis and Pre-Processing

We began with a preliminary exploratory analysis, which revealed a marked class imbalance. To ensure a representative split, we stratified the data into training and validation sets (80%, with 20% of this subset reserved for validation) and a separate testing set (10%). Images were resized to 224×224 pixels and normalised using the '*preprocessing_input*' function to ensure compatibility with pre-trained weights and promote training stability.

To enrich the dataset and mitigate overfitting, we applied moderate augmentations, including rotations, zooms, and horizontal flips, exclusively to the training set. We also tested additional preprocessing strategies, such as grayscale conversion, background removal, circular masking, and peripheral blurring. As these approaches did not lead to noticeable improvements, we opted to retain the images close to their original state.

During visual inspection, we identified and manually removed irrelevant images, such as paper sheets and maps, resulting in a clean training set used for all subsequent procedures.

2.3 Baseline Model

In order to get an initial idea of how our model would perform, we started by implementing a well-known and conventional model, the *CNN*. The goal of this implementation was to establish a baseline score, and through this, we were able to have a direction and focus to improve with more complex and more advanced models for image classification.

To extract the primary features of the input images, the architecture of the baseline model consisted of three convolutional blocks, each of which was followed by a maximum pooling layer. After extracting the features, a dense layer with 128 neurons and Relu activation flattened and processed the representations. To reduce the possibility of overfitting, a dropout layer was then used. Finally, for multi-class classification issues, the output layer uses the softmax activation function to output a probability distribution across the target classes.

2.4 Models

After establishing a baseline, we implemented more robust models, including *EfficientNetB0*, *ResNet50*, *DenseNet121*, *MobileNetV2* and *VGG16*. By loading pre-trained weights into ImageNet, all of the models employed Transfer Learning, which greatly enhanced performance.

Given the performance improvement over simpler models, EfficientNetB0, the first transfer learning model we evaluated, was probably the one that attracted the most attention. All of the weights were fully trained when we used the model with '*include_top=False*' and '*weights="imagenet"*'. To create the probabilities of each class, we first applied a GlobalAveragePooling2D layer, which condenses each feature map into a single average value. This was followed by a dense layer with 512 neurons, Dropout of 0.5 and 0.3, and an output layer with softmax activation. We employed EarlyStopping to prevent overfitting by terminating training when the validation loss stopped improving, ReduceLROnPlateau to decrease the learning rate during periods of stagnation, and the Adam optimiser to train because of its capacity to dynamically modify the learning rate. Additionally, we recorded the training history using CSVLogger and adjusted for class imbalance using the class_weight parameter.

MobileNetV2 was another model that we focused on due to its performance. To reduce the dimensionality of the feature maps, we decided to apply 'p*ooling=* "*avg*"' and eliminate the initial classification layer. Utilising the previously learnt weights, the base was maintained frozen. To enhance generalisation, we inserted dense layers with 512 and 256 neurons, separated by Dropout, after feature extraction. BatchNormalization was also used to speed up and stabilise training. The probability distribution across the classes was generated by the last layer using softmax activation, same like in the other models.

Additionally, pre-trained ImageNet weights and frozen bases were employed with the *ResNet50*, *DenseNet121*, and *VGG16* models. Combining layers of BatchNormalization, dense layers with 512 and 256 units, Dropout for regularisation, and a final layer with softmax activation, the top structure used was comparable for both. *DenseNet121* links its layers densely, encouraging feature reuse, whereas *ResNet50* uses residual blocks to let gradients flow. The more traditional *VGG16* architecture, which is built on basic convolutional blocks, has good performance but is very slow and due need to explore the model using gridsearch, we decided not to use it, but it is nevertheless useful as a benchmark for contrasting with more modern models.

3. Results and Evaluation

3.1 Metrics

To estimate the performance of the different models run throughout the project, we relied on a set of complementary measurements. AUC, accuracy, and training/validation loss were used to quantify the overall quality of the predictions and monitor the model's learning pattern over time so that we can identify early indicators of overfitting or underfitting. Additionally, we employed precision, recall, F1-score, and a confusion matrix to give a more detailed class-wise analysis. These metrics were particularly important given the imbalanced nature of the dataset, as they informed us about which families were harder to label and identify dominant patterns of mislabeling.

3.2 Best Model Performance

After testing various models and researching the most suitable models for image classification, we came to the conclusion that the model that gives us the best performance is *EfficientNetB0* with hyper tuning, using the pre-trained ImageNet weights. During testing, we obtained values of 0.5713 for accuracy, 1.784 for loss and 0.9502 for AUC, which shows a very reasonable performance. Despite the good performance, the model contains some overfitting, since it has an accuracy of 0.7817 in training, something we think may have been caused by the imbalance of the classes and also by the complexity of the data.

Through hyperparameter tuning, we developed an adjusted version with the goal of optimizing the model's performance. This technique involved automatically defining the best values for the most important architectural hyperparameters, such as the number of neurons in dense classrooms, dropout rates, and learning rates. The resulting model used the pre-treated ImageNet weights, maintained the congelated base, and added dense images with the appropriate dimensions and configurations found during the application process. This adjustment was crucial to improving the model's generalization ability, reducing overfitting, and better tailoring the architecture to the unique characteristics of our data set.

3.3 Error Analysis

We analysed the predictions on the validation and test sets by generating a confusion matrix and inspecting misclassified examples. The most common cases of misclassification are from similar species across different families that lacked images with clear distinguishing features (e.g. colour, shape, or background).

Visual Similarity Between Classes: Within the same phylum, families that shared significant morphological features would lead to frequent misclassifications.

Classes with Low Sample Data: Despite the use of class-weight to handle the imbalance of the data, classes with few images are prone to underperform.

Background Noise: Even with preprocessing strategies that revolved around removing the background, the images would still present persons or human limbs, and sometimes objects in focus, which would not be removed.

Model Overfitting: Between training and validation, the accuracy presents a substantial gap, which indicates that the model has memorised the training patterns and has trouble predicting unseen data. This overfit may be caused by the removal of outlier images during training, but is still present in validation and test.

4. Future Work

Although our final model achieved solid performance, several improvements could be explored in future work:

- Overfitting Reduction: We would test more sophisticated augmentations (like mixup or cutoff), stronger regularisation (like L2 weight decay), and model ensembling for improved generalisation in order to address the observed overfitting.
- Automatic Filtering of Non-Animal Images: It was ineffective to manually eliminate unnecessary images (such as text or maps). This cleaning procedure could be automated using an OCR-based detection system or a lightweight picture filter.
- Multimodal Learning: Using multimodal fusion, adding metadata (such as phylum or kingdom) to image data may enhance classification, especially for visually related classes.
- Other Architectures: Better results might be obtained by experimenting with Vision Transformers or fine-tuning deeper EfficientNet variations.
- Class Imbalance Handling: Future research could investigate data augmentation or targeted loss strategies aimed at minority categories to enhance predictions on uncommon classes.

5. Conclusion

After establishing a baseline model using a basic *CNN*, we investigated a series of pre-trained architectures with preprocessing and optimisation techniques, such as EfficientNetB0, *ResNet50, DenseNet121, MobileNetV2*, and *VGG16*. Despite still showing signs of overfitting, the *EfficientNetB0* model was the best-performing option, with respectable generalisation capacity. Recurring constraints, including visually identical classes, classes with few examples, and background noise in the photos, were found through error analysis. We are confident that in the next iteration of the project, in which we can have greater computational power, given that this was the great difficulty and challenge we faced, we will certainly mitigate many of the "problems" that our implementation contains. But despite these difficulties, the results show the promising potential of these methods for classifying biodiversity.

Annexes

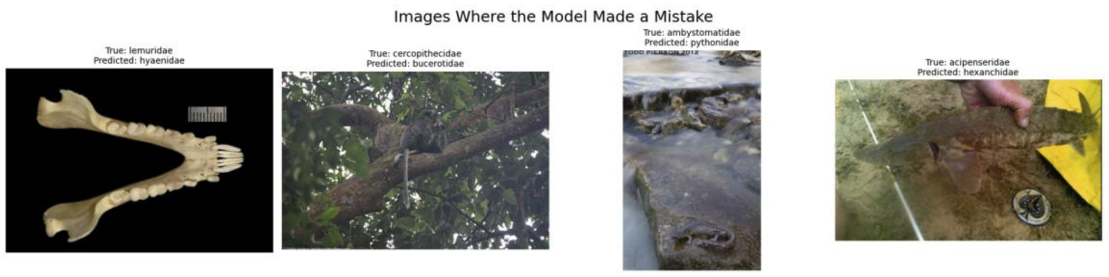

Figure 1 Example of test images misclassified by EfficientNetB0

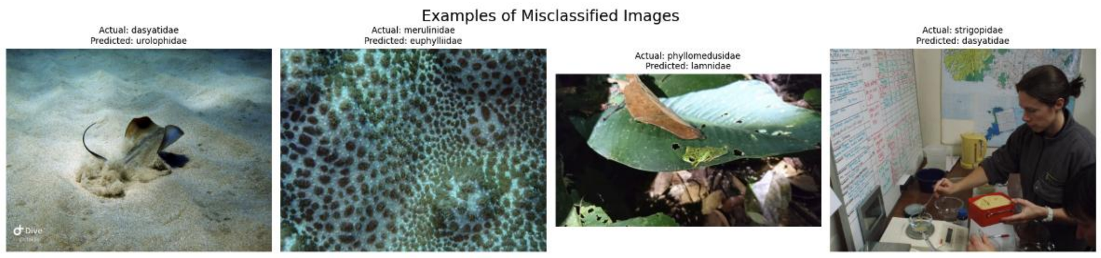

Figure 2 Example of test images misclassified by MobileNetV2

References

Python Software Foundation. (n.d.). *os — Miscellaneous operating system interfaces*. Python 3 documentation. Retrieved April 30, 2025, from https://docs.python.org/3/library/os.html#os.listdir

Matplotlib. (n.d.). *Image tutorial*. Retrieved April 30, 2025, from https://matplotlib.org/stable/tutorials/images.html

Pillow Developers. (n.d.). *Image module*. Pillow (PIL Fork) Documentation. Retrieved April 30, 2025, from https://pillow.readthedocs.io/en/stable/reference/Image.html

MathWorks. (n.d.). *Process data for deep neural networks*. Retrieved April 30, 2025, from https://www.mathworks.com/help/deeplearning/process-data-for-deep-neural-networks.html

ReginaExMachina. (n.d.). *Animal detection model* [GitHub repository]. Retrieved April 30, 2025, from https://github.com/ReginaExMachina/animal-detection-model

Built In. (n.d.). *What is transfer learning?*. Retrieved April 30, 2025, from https://builtin.com/data-science/transfer-learning

Fagbemi, D. (2023, April 13). *Guide to transfer learning in deep learning*. Medium. Retrieved April 30, 2025, from https://medium.com/@davidfagb/guide-to-transfer-learning-in-deep-learning-1f685db1fc94