DEEP LEARNING PROJECT

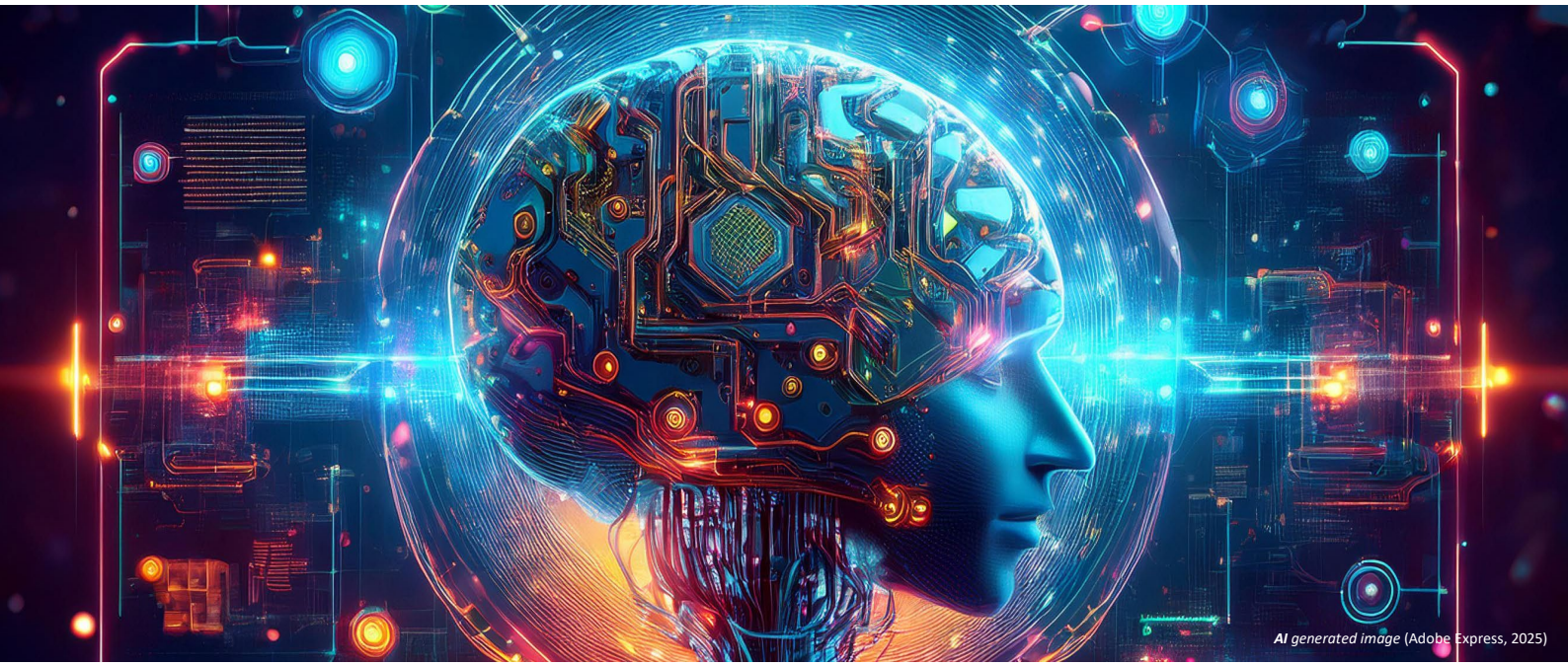
Master in Data Science and Advanced Analytics

NOVA Information Management School

Universidade Nova de Lisboa

Predicting Rare Species from Images

using Deep Learning



Group 37

André Silvestre, 20240502

Diogo Duarte, 20240525

Filipa Pereira, 20240509

Maria Cruz, 20230760

Umeima Mahomed, 20240543

Spring Semester 2024-2025

TABLE OF CONTENTS

ABSTRACT

This project developed a deep learning model to classify rare species images from the BioCLIP dataset into their respective families. Addressing class imbalance through SMOTE-inspired data augmentation and comparing baseline CNNs against pre-trained models (*VGG19*, *ResNet152V2*, *EfficientNetV2B0*, *ConvNeXtBase*), *ConvNeXtBase* with SMOTE augmentation emerged as the top performer. An innovative approach used the CLIP model for zero-shot classification to remove non-animal images (~15% of the data). The final model, trained again on the filtered dataset, reached 83.1% accuracy and 78.7% Macro F1-Score on the test set. This work offers a reliable automated method for rare species family classification, supporting biodiversity conservation.

Keywords: Rare Species Image Classification; Deep Learning; CNN; Transfer Learning; Data Augmentation; CLIP

---

1. INTRODUCTION

Identification of rare species is crucial for maintaining biodiversity but often relies on time-consuming expert analysis. Deep Learning (DL) and particularly Convolutional Neural Networks (CNNs), provides a solution for automatic species identification directly from images, overcoming issues like visual similarity and insufficient data [1]. This project uses DL to build a model that classifies rare species into families based on images. Sourced from the BioCLIP dataset via the Encyclopedia of Life (EOL) [2], images and metadata (kingdom, phylum, family) are the foundation required to train family-level classification model.

Recent studies prove that CNNs, usually improved by transfer learning with models like *ResNet*, *EfficienitNet*, and *VGG*, achieve high accuracy in classifying diverse taxa such as birds, mammals, and sea life [3][4][6][9][11][16]. Techniques like data augmentation [5][7][11][14] and models like *YOLO* or *R-CNN* [8][10] address challenges in image quality and background problems, particularly for camera trap images [9][16][19][21]. This shows that DL offers robust tools for tracking and conserving biodiversity, guiding the practices undertaken in our project.

This report outlines how our species classification model was developed. **Section 2** presents data exploration and image preprocessing. **Section 3** explains the modelling approach selected, experimental setup, and model performance analysis using relevant metrics. **Section 4** explores an innovative technique. Finally, **Section 5** summarizes main findings, limitations, and suggests future research directions.

2. DATA EXPLORATION AND IMAGE PREPROCESSING

An initial exploratory analysis of the provided metadata CSV file using the *pandas* library confirmed the dataset contains 11,983 observations and 7 features. No missing values or duplicate rows were detected, and all data types suited their features.

Metadata analysis showed that all images belong to *kingdom Animalia*, making this a non-discriminatory attribute in our classification problem. The *phylum* attribute has 5 unique values, of which 83% are Chordata (*vertebrates*), as shown in **Figure B1**. The target variable, *family*, comprises 202 unique classes. The frequency distribution between these families is notably imbalanced, ranging from 29 samples for the least frequent family (*Siluridae*) to 300 for the most frequent (*Dactyloidae* and *Cercopithecidae*), illustrated in **Figure B2**. A hierarchical check ensured that each family was placed only in a unique phylum. Moreover, a brief visual inspection of images (**Figure B3**) indicated that some images are not representative of animals (e.g., showing habitat or specimen labels), representing potential outliers that could affect model training.

Based on literature review [4][6][8][14][15], the dataset was split into training (80%), validation (10%), and test (10%) sets. This split was performed using stratification to maintain the original proportion of each family within the sets, mitigating potential biases due to the class imbalance. Shuffling was applied during the split. For practical workflow management between multiple notebooks (sharing common functions via a *utilities.py* file),

the image files were copied into three separate folders. *Hold-out method* was chosen over *k-fold cross-validation* primarily based on computational costs. The validation set was used for hyperparameter tuning, while the test set was kept aside to evaluate the model's performance and its generalization capacity.

To address the class imbalance in the target variable, we considered *oversampling*, *undersampling*, and *class weighting*. *Undersampling* wasn't appropriate as it would discard many of images from the majority classes, leading to significant data loss, especially given the already limited size of the dataset for some families. Therefore, we decided to explore *oversampling* (using data augmentation) and *class weighting* during the modelling phase.

2.1. Image Import & Preprocessing

The *Keras* library [25] within *TensorFlow (tf)* [26] was used to data loading, preprocessing, and modelling. The images were loaded from the *image_dataset_from_directory* function, configured as follows:

- **Image Size:** (224, 224) pixels, which is common input size for most pre-trained models [27][28][29][30] and common literature practices [6][9][11][14][17][18][20].
- **Colour Mode:** *rgb* (3 channels) was used to retain full colour information.
- **Aspect Ratio Management:** *crop_to_aspect_ratio=False* and *pad_to_aspect_ratio=False*. Cropping can remove important features, and padding can introduce misleading information (black bars), as illustrated in **Figure B4**. Non-uniform scaling, despite potential distortion, was preferred to preserve most original pixel data.
- **Labels:** *label_mode='categorical'* generated one-hot encoded (OHE) labels for compatibility with categorical cross-entropy loss and *class weighting*. *labels='inferred'* utilized the subdirectory names as labels.
- **Shuffling:** *shuffle=True* applied only to the training set import, crucial for breaking data order and improving model generalization during batch processing. Validation and test sets used *shuffle=False*.
- **Batch Size:** *64*, determined as the largest feasible size given hardware constraints.
- **Interpolation:** *'bilinear'* provided a balance between resizing quality and computational cost.

Before actually going for preprocessing steps, the image transformations available in *tf.image* were tried out visually (**Figure B5**). Based on the idea of isolating the animal from the background, we decided to try out systematically applying *grayscale*, *adjust_contrast*, and *adjust_saturation* (with factor of 1.5) individually in the baseline model evaluation phase.

2.2. Addressing Imbalance Issues

Two methods were employed to minimize the class imbalance effect:

Oversampling via Data Augmentation ("SMOTE"): Instead of traditional SMOTE which operates in the feature space (not directly applicable to images), we implemented an *oversampling* strategy by applying data augmentation to the minority classes in the training set. A copy of the training data folder was created. For each class with fewer samples than the largest class (n=240), new images were generated using Keras's *RandAugment* layer [31][32] applied to existing images until the sample count matched the largest class. *RandAugment* applies a sequence of randomly selected augmentation techniques (e.g., rotation, colour – **Figure B6**) with random magnitudes, introducing significant diversity. While less effective than real-time augmentation, this approach provided consistent results, as the same SMOTE-augmented dataset was used for all model training in the SMOTE condition.

Class Weighting: The *class_weight* parameter was also tried out when training the model. Weights were calculated for each class in inverse proportion to their frequency based on the formula:

$$\text{weight\_for\_class\_i} = \frac{\text{total\_samples}}{\text{num\_classes} \times \text{samples\_in\_class\_i}}$$

This approach increases the influence of misclassifications from rarer classes to the loss function, forcing the model to pay more attention to them while optimizing.

3. MODELLING & EVALUATION

The modelling process followed the strategy outlined in **Figure 3.1**, starting with a baseline CNN, evaluating combinations of preprocessing and imbalance handling, selecting promising combinations, testing these on pre-trained models, and finally fine-tuning the best overall model.

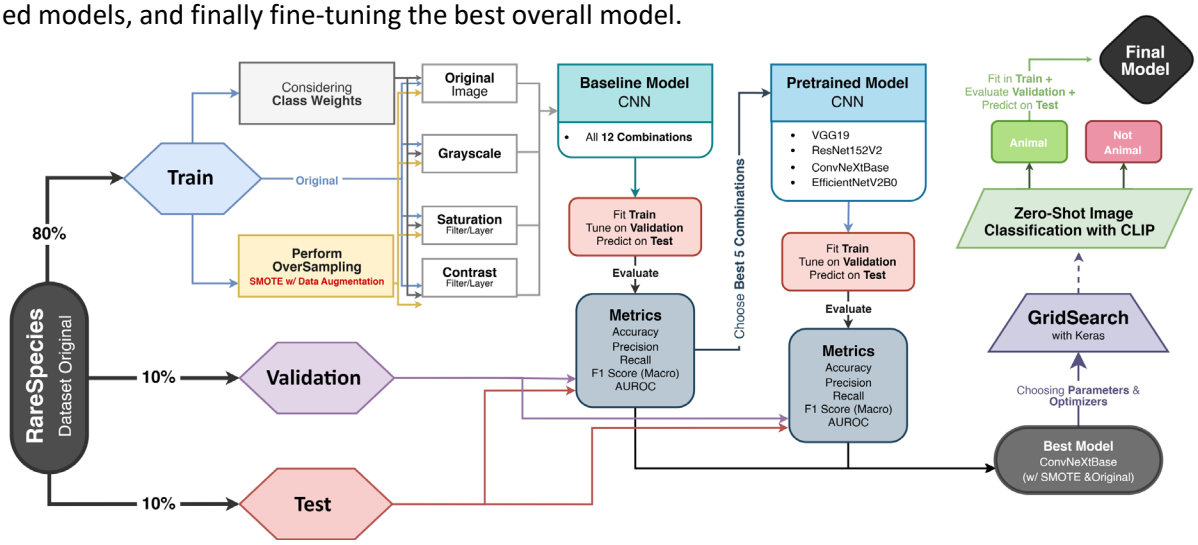


Figure 3.1 – Model Process Flowchart**.**

3.1. Baseline Model – Combination Selection

We built a baseline CNN model using the *Keras* functional *API* (**Figure C1**). It starts with a **Rescaling** layer to normalize pixel values to [0, 1] before specific image preprocessing. The model includes blocks of **Conv2D** layers (with 32, 64, 128, and 256 filters), **BatchNormalization**, and *ReLU* **activation**, followed by **MaxPooling2D** to reduce dimensions. Conv2D layers capture spatial features, BatchNormalization stabilizes training, and MaxPooling lowers computational load. A **GlobalAveragePooling2D** layer flattens the feature maps, followed by a **Dense** layer (128 units, *ReLU* activation) with **BatchNormalization**. A **Dropout** layer (0.5 rate) prevents overfitting, and a final **Dense** layer (202 units, *softmax* activation) outputs probabilities for the 202 classes.

This baseline model was used to evaluate different combinations of image preprocessing (original RGB, grayscale, contrast, saturation) and imbalance handling techniques (original data, "SMOTE" via augmentation, class weights). These initial runs were limited to a maximum of 10 epochs (**Table C1**) to quickly identify promising combinations, acknowledging that models might not fully converge.

The primary metric for evaluation was the *Macro F1-Score* (MF1), supplemented by *Accuracy*, *Precision*, *Recall*, and *AUROC* Score. Since the goal is to classify rare species, *MF1* is the best metric choice as it averages the F1 score across all classes, giving equal importance to each family, regardless of its size. Poor performance on some classes like the rarer ones is indicated by low MF1. While *Accuracy* gives an overall correctness measure, when alone, it can be misleading with imbalanced data. *Precision* and *Recall* help diagnose error types (FP/FN), and *AUROC* measures the model's ability to distinguish between classes across different thresholds.

We used standard *Keras* callbacks during training. **ModelCheckpoint** saved the best model based on validation loss. **CSVLogger** recorded metrics for each epoch for later review. **LearningRateScheduler** was included to adjust the learning rate with a decay schedule. **EarlyStopping** finished training if the validation loss stopped improving after 3 epochs, helping avoid overfitting and unnecessary training.

The **Adam** optimizer [33] was chosen (*learning_rate*=0.001, *weight_decay*=0.01) due to its adaptive learning rate and generally fast convergence. The **CategoricalCrossentropy** loss function [1] was applied, given our multi-class classification problem with OHE labels.

Based on the initial 10-epoch results (**Table C1**), combinations involving grayscale, class weights, and original images with only saturation adjustment consistently showed lower performance across most metrics. The

grayscale combinations probably performed poorly due to the removal of the ability to distinguish between the colours of different species. Likewise, the *class weighting* might have not yielded expected results, probably needing more tuning or longer training.

The five most promising combinations (Original, Original+Contrast, SMOTE, SMOTE+Contrast, SMOTE+Saturation) were then retrained using the baseline architecture, allowing up to 100 epochs. It was expected that these models might overfit the training data as the primary goal here was to compare them and choose the best one for later fine-tuning.

3.2. Pretrained Models | Transfer Learning

To use knowledge learned from large datasets like ImageNet, we applied transfer learning with pre-trained models from *Keras Applications* [24]. **Table A1** provides further details on each architecture. The expectation was that these models, with their optimized architectures, would achieve better performance and faster convergence compared to the baseline CNN. Models were selected based on the literature review, performance benchmarks, and compatibility with the (224, 224) input size.

An essential step when using these models is applying the specific *preprocess_input* layer associated with each architecture. These layers handle normalization and potential colour conversions (e.g., RGB to BGR for VGG) according to how the models were originally trained on ImageNet. Failure to use the correct preprocessing layer significantly weakens the benefits of transfer learning.

The five selected combinations were applied to each of the four pre-trained models (VGG190 [27], ResNet152V2 [28], ConvNeXtBase [29], EfficientNetV2B0 [30]), freezing the base layers and adding a similar classification head as the baseline (**GlobalAveragePooling**, **Dense**, **Dropout**, **Output Dense**). These models were trained for up to 100 epochs with the same callbacks, optimizer, and loss function as before.

3.3. Evaluation Models

In nearly all experiments, the learning curves, as observed through the loss (which can be consulted on the notebooks), demonstrate that the models are training successfully.

Analysing the results from **Table C2**, particularly focusing on the validation MF1 and considering overfitting (difference between training and validation Accuracy and MF1), the *ConvNeXtBase* model with the SMOTE (augmentation) strategy and no additional colour preprocessing (Contrast/Saturation) demonstrated the best overall performance. It achieved the highest validation (0.804) and test MF1 (0.829) scores, with tolerable overfitting (Train MF1: 0.886), therefore, in the fine-tuning phase we plan to address this overfitting.

3.4. Fine-tuning

As described above, we proceeded with fine-tuning through Keras Tuner, ultimately selecting the *Hyperband* strategy [36] over *RandomSearch* [35] and *GridSearch* [34] for its efficiency in large search spaces (**Table B1**). The search aimed to optimize the validation Macro F1-Score over a maximum of 10 epochs.

Several aspects were tuned: the possibility of **unfreezing** the final 22 layers of the *ConvNeXtBase* model, while keeping Batch Normalization layers frozen as is standard practice [37], to allow adaptation of higher-level features; selecting an appropriate **optimizer** (*adam, sgd, rmsprop*); adjusting the **learning rate** to lower values, recognizing that these are often necessary when unfreezing pre-trained layers; and modifying the **dropout rate** in the custom classification head to manage regularization. (**Table D1**)

Subsequently, the model was rebuilt using these best hyperparameters and retrained up to 100 epochs and employed all previous callbacks adding *ReduceLROnPlateau* [38] to dynamically adjust the learning rate in an attempt to boost convergence and performance.

Despite this attempt (due to memory limitations we were only able to run 7 trials), the results did not improve, leading us to explore an innovative approach while maintaining the previously defined parameters.

4. INNOVATIVE APPROACH

During data exploration (**Figure B3**), it was noted that some images in the dataset did not contain identifiable animals, instead showing specimen labels, habitat shots, or other non-target subjects. These images act as outliers and could potentially introduce unnecessary noise.

To address this, a Zero-Shot Image Classification [39] approach using the CLIP (Contrastive Language–Image Pre-training) model was implemented. CLIP models can classify images based on natural language descriptions without having been explicitly trained on the target classes. We used the *clip-vit-base-patch16* model [40] via the *Hugging Face transformers* library [41][42], chosen for its balance between performance and computational feasibility. Due to version compatibility issues between *Transformers* and *Keras* used before, this classification step was performed in a separate *PyTorch* environment. Examples of the CLIP model's classification output are shown in **Figure E1** where each image was classified as *"animal"* or *"not animal"*.

The model demonstrated a reasonable ability to distinguish between images containing animals and those without. Applying this classification across the dataset revealed that approximately 15% of images in each split were classified as *"not animal"* (**Figure E2**), without compromising the proportion of each split.

Our **final *ConvNeXtBase* model**, trained on the CLIP-filtered and SMOTE-augmented dataset, did well on the unseen test set, reaching **83.1%** accuracy and a **78.7%** Macro F1-Score (**Table F1**). This shows most test images were correctly classified, with balanced performance across common and rare families. Learning curves (**Figure F1**) indicate good generalization, as validation metrics followed training closely until early stopping. The gap between train and validation/test results stayed below 5%, showing the model handles new data effectively. The confusion matrix (**Figure F4**) has a clear diagonal pattern, confirming accurate classification of most of the 202 families. Examining prediction examples (**Figure F2** and **Figure F3**) revealed errors often involved species with similar shapes, colours, or unclear backgrounds. Low contrast, camouflaged subjects, or juvenile forms differing from adults also caused issues. Human hands in images or possible dataset labelling errors may have also contributed to some misclassifications.

5. CONCLUSION

This project effectively utilized DL to classify rare species into families using the *BioCLIP* dataset. Initial exploration of the data revealed class imbalance and the presence of images lacking animals. Preprocessing was conducted, involving normalized image resizing, image transformations and addressing imbalance through augmentation-based *oversampling* and experiments with *class weighting*.

A baseline CNN and four pre-trained models were assessed across various setups, resulting in 12 baseline candidates and 25 trained model configurations. The *ConvNeXtBase* architecture, paired with SMOTE augmentation and its native *preprocess_input* layer, was found to deliver the best performance, as measured by the *Macro F1-Score* and *Accuracy*. However, fine-tuning this model did not lead to improved results.

An innovative approach employing CLIP for zero-shot classification was implemented, successfully identifying approximately 15% of images as likely non-animal subjects. Retraining on the filtered *"OnlyAnimals"* dataset produced more robust (less overfitting) results with slightly higher accuracy.

This project faced challenges due to limited data for very rare families and computational limits that restricted the extent of hyperparameter tuning. Future work could include using k-fold cross-validation, applying advanced augmentation or generative methods for *oversampling*, incorporating metadata features, and potentially expanding the dataset through web scraping or other sources to address these limitations.

Overall, this project corroborates the importance of transfer learning with advanced CNN architectures for challenging visual classification tasks, such as identifying rare species, highlighting the importance of careful preprocessing, addressing class imbalance, and thoughtful fine-tuning.

BIBLIOGRAPHICAL REFERENCES

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. The MIT Press.

[2] Stevens, S., Wu, J., Thompson, M. J., Campolongo, E. G., Song, C. H., Carlyn, D. E., ... & Su, Y. (2024). Bioclip: A vision foundation model for the tree of life. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 19412-19424).

[3] S, N. J., Kamsala Tharun, & Somu Geetha Sravya. (2024). Deep Learning Approaches to Image-Based Species Identification. *2024 International Conference on Integrated Circuits and Communication Systems (ICICACS)*, 1–7. https://doi.org/10.1109/icicacs60521.2024.10498423

[4] Bhargavi, I., Pratap, A. R., & Sri, A. S. (2024). An Enhanced EfficientNet-Powered Wildlife Species Classification for Biodiversity Monitoring. *2024 4th International Conference on Intelligent Technologies (CONIT)*, 1–6. https://doi.org/10.1109/conit61985.2024.10627148

[5] Mane, V., Pranjali Nikude, Patil, T., & Tambe, P. (2024). Wildlife Classification using Convolutional Neural Networks (CNN). *2024 International Conference on Inventive Computation Technologies (ICICT)*. https://doi.org/10.1109/icict60155.2024.10544702

[6] Habib, S., Ahmad, M., Ul Haq, Y., Sana, R., Muneer, A., Waseem, M., … Dev, S. (2024). Advancing Taxonomic Classification Through Deep Learning: A Robust Artificial Intelligence Framework for Species Identification Using Natural Images. *IEEE Access*, *12*, 146718–146732. doi:10.1109/ACCESS.2024.3450016

[7] Kimly Y, Malis Lany, Soy Vitou, & Kor, S. (2023). Animal Classification using Convolutional Neural Network. *The 2nd Student Conference on Digital Technology 2023*. https://www.researchgate.net/publication/376751387_Animal_Classification_using_Convolutional_Neural_Network

[8] Sharma, S., Sisir Dhakal, & Bhavsar, M. (2024). Transfer Learning for Wildlife Classification: Evaluating YOLOv8 against DenseNet, ResNet, and VGGNet on a Custom Dataset. *Journal of Artificial Intelligence and Capsule Networks*, *6*(4), 415–435. https://doi.org/10.36548/jaicn.2024.4.003

[9] Supreet Parida, Mishra, A., Sahoo, B. P., Nayak, S., Mishra, N., & Panda, B. S. (2024). Recognizing Wild Animals from Camera Trap Images Using Deep Learning. *2024 International Conference on Intelligent Computing and Emerging Communication Technologies (ICEC)*, 1–6. https://doi.org/10.1109/icec59683.2024.10837421

[10] aa Gagandeep M D, Jagath S K, Kartik Tomar, Senthil Kumar R. (2024). R-CNN Based Deep Learning Approach for Counting Animals in the Forest: A Survey. *International Journal of Networks and Systems*, *13*(1), 1–4. https://doi.org/10.30534/ijns/2024/011312024

[11] Pruthvi Darshan S S, L, J. M., & Sangeetha V. (2024). Multiclass Bird Species Identification using Deep Learning Techniques. *2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 1–6. https://doi.org/10.1109/conecct62155.2024.10677184

[12] Gill, K. S., Gupta, R., Malhotra, S., Swati Devliyal, & G Sunil. (2024, April 5). Classification of Reptiles and Amphibians Using Transfer Learning and Deep Convolutional Neural Networks. *2022 IEEE 7th International Conference for Convergence in Technology (I2CT)*. https://doi.org/10.1109/i2ct61223.2024.10544030

[13] P Kanaga Priya, T Vaishnavi, N Selvakumar, G Ramesh Kalyan, & A Reethika. (2023, July 19). An Enhanced Animal Species Classification and Prediction Engine using CNN. *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*. https://doi.org/10.1109/icecaa58104.2023.10212299

[14] Sharma, S., Neupane, S., Gautam, B., & Sato, K. (December 2023). Automated Multi-Species Classification Using Wildlife Datasets Based on Deep Learning Algorithms. *Materials, Methods & Technologies*, *17*, 2023. doi:10.62991/MMT1996359772

[15] Oion, M. S. R., Islam, M., Amir, F., Ali, M. E., Habib, M., Hossain, M. S., & Wadud, M. A. H. (2023). Marine Animal Classification Using Deep Learning and Convolutional Neural Networks (CNN). *2023 26th International Conference on Computer and Information Technology (ICCIT)*, 1–6. doi:10.1109/ICCIT60459.2023.10441585

[16] Binta Islam, S., Valles, D., Hibbitts, T. J., Ryberg, W. A., Walkup, D. K., & Forstner, M. R. J. (2023). Animal Species Recognition with Deep Convolutional Neural Networks from Ecological Camera Trap Images. *Animals*, *13*(9), 1526. https://doi.org/10.3390/ani13091526

[17] Cai, R. (2023). Automating bird species classification: A deep learning approach with CNNs. *Journal of Physics: Conference Series*, *2664*(1), 012007. https://doi.org/10.1088/1742-6596/2664/1/012007

[18] Priya, P. K., Vinu, M. S., PrasannaBlessy, M., Kirupa, P., Gayathri, R., & Selvakumar, N. (2023). An Eagle-Eye Vision: Advancements in Avian Species Classification. *2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS)*, 758–764. doi:10.1109/ICACRS58579.2023.10404897

[19] Larson, J. (2021). *Assessing Convolutional Neural Network Animal Classification Models for Practical Applications in Wildlife Conservation* [MSc Thesis]. https://doi.org/10.31979/etd.ysr5-th9v

[20] Sanghvi, K., Aralkar, A., Sanghvi, S., & Saha, I. (May 2020). *Fauna Image Classification using Convolutional Neural Network*. *13*, 8–16.

[21] Rajasekaran, T., Kaliappan, V., Surendran, R., Sellamuthu, K., & Palanisamy, J. (October 2019). Recognition Of Animal Species On Camera Trap Images Using Machine Learning And Deep Learning Models. *International Journal of Scientific & Technology Research*, *8*.

[22] Albuquerque, C. (2019). Convolutional neural networks for cell detection and counting : a case study of human cell quantification in zebrafish xenografts using deep learning object detection techniques [MSc Thesis]. https://run.unl.pt/handle/10362/62425

[23] Team, K. (n.d.-b). *Keras documentation: LearningRateScheduler*. https://keras.io/api/callbacks/learning_rate_scheduler/

[24] Team, K. (n.d.). *Keras documentation: Keras Applications*. https://keras.io/api/applications/

[25] Chollet, F., et al. (2015). Keras. https://keras.io

[26] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., & Levenberg, J. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. ArXiv.org. https://arxiv.org/abs/1603.04467

[27] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. Arxiv. ICLR. https://doi.org/10.48550/arXiv.1409.1556

[28] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity Mappings in Deep Residual Networks. Arxiv. https://arxiv.org/pdf/1603.05027

[29] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., Xie, S., Facebook, A., & Research. (2022). *A ConvNet for the 2020s*. https://arxiv.org/pdf/2201.03545

[30] Tan, M., & Le, Q. V. (2021). EfficientNetV2: Smaller Models and Faster Training. Arxiv.org. https://doi.org/10.48550/arXiv.2104.00298

[31] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data, 6*(1), 1-48. https://doi.org/10.1186/s40537-019-0197-0

[32] Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). RandAugment: Practical automated data augmentation with a reduced search space. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 702-703. https://arxiv.org/abs/1909.13719

[33] Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization.* https://arxiv.org/pdf/1412.6980

[34] Claesen, M., & De Moor, Bart. (2015). *Hyperparameter Search in Machine Learning*. ArXiv.org. https://arxiv.org/abs/1502.02127

[35] Bergstra, J., & Bengio, Y. (2012). *Random Search for Hyper-Parameter Optimization.* Journal of Machine Learning Research, 13, 281–305.

[36] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*. Journal of Machine Learning Research, 18(185), 1–52.

[37] Team, K. (2020). *Keras documentation: Transfer learning & fine-tuning*. Keras.io. https://keras.io/guides/transfer_learning/#finetuning

[38] Team, K. (n.d.). Keras documentation: ReduceLROnPlateau. Keras.io. https://keras.io/api/callbacks/reduce_lr_on_plateau/

[39] Hugging Face. (2022, October 4). *What is Zero-Shot Image Classification? - Hugging Face*. Huggingface.co. https://huggingface.co/tasks/zero-shot-image-classification

[40] OpenAI. (2021). *openai/clip-vit-base-patch16*. *Hugging Face Model Hub.* https://huggingface.co/openai/clip-vit-base-patch16

[41] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020*). Transformers: State-of-the-art natural language processing.* Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 38-45.

[42] Hugging Face. (2024). *Hugging Face – On a mission to solve NLP, one commit at a time.* Huggingface.co. https://huggingface.co/

APPENDIX A. LITERATURE REVIEW

Table A1 – Literature Review

(Chronologically ordered from the most recent article to the oldest)

Paper Title	Abstract Summary	Methodology	Main Findings	Image Preprocessing Techniques Used	Algorithms Used	Reference
Deep Learning Approaches to Image-Based Species Identification	Explores DL and transfer learning for automating image-based species identification from images, contributing to biodiversity conservation efforts.	- Used TensorFlow, Keras, and OpenCV; - Trained multiple CNN models; - Applied data augmentation and optimization techniques (EarlyStopping and learning rate adjustments); - Compared transfer learning models; - Fine-tuned Xception.	Transfer learning with Xception was most effective.	Data Augmentation, Image processing with OpenCV	ResNet50V2 ResNet152V2 InceptionV3 Xception DenseNet121	[3]
An Enhanced EfficientNet-Powered Wildlife Species Classification for Biodiversity Monitoring	Focuses on an EfficientNet model for efficient and accurate wildlife species classification, supporting biodiversity monitoring.	- Used EfficientNet architecture (transfer learning); - Used advanced preprocessing methods (Lanczos interpolation); -Incorporated data augmentation and attention mechanisms.	The model showed efficiency and accuracy (useful for biodiversity monitoring).	Feature Normalization, Transfer Learning, Data Augmentation, Attention Mechanisms	EfficientNet	[4]
Wildlife Classification using Convolutional Neural Networks (CNN)	Develops a CNN-based model for accurate identification and classification of diverse wildlife species, contributing to wildlife conservation efforts.	-Developed a CNN model; - Used high-resolution wildlife images; - Optimized hyperparameters; - Utilized data augmentation; - Fine-tuned the model; - Evaluated accuracy and validation loss on a separate test dataset.	Achieved precise classification, supporting wildlife conservation applications.	Data Augmentation	CNNs	[5]

Advancing Taxonomic Classification Through Deep Learning: A Robust Artificial Intelligence Framework for Species Identification Using Natural Images	A DL framework using ResNet-50 CNN achieves high accuracy in classifying species, including rare animals, from natural images.	-Utilized ResNet-50; - Pre-processed and augmented the dataset; - Modified ResNet-50 for 4 million trainable parameters; - Compared with models like GoogleNet, VGG, SegNet, and DeepLab v3+.	ResNet-50 modified showed superior computational efficiency and accuracy compared to existing models.	Grayscale Conversion Data Augmentation (Rotation)	ResNet-50 CNN Modified ResNet-50	[6]
Animal Classification Using CNN Architectures	Explores the use of CNN architectures for classifying wild animal species, contributing to conservation efforts for a dataset of 90 animal images from Kaggle.	- Used CNNs like LeNet-5, AlexNet, VGG16, ResNet50, Inception V3, and InceptionResnetV2 and implemented data augmentation; - Used dropout to prevent overfitting.	Compared performance of different CNNs, with InceptionResnetV2 performing best.	Data Augmentation	LeNet-5, AlexNet, VGG16 ResNet50 Inception V3 InceptionResnetV2	[7]
Transfer Learning for Wildlife Classification: Evaluating YOLOv8 against DenseNet, ResNet, and VGGNet on a Custom Dataset	Evaluates the performance of DL models, including YOLOv8, for classifying rare wildlife species using a custom dataset.	80/20 train-validation split; - Fine-tuned pre-trained models by freezing initial layers and adding custom fully connected layers; - Used weighted Adam optimizer and categorical cross-entropy loss function for 100 epochs with a batch size of 32.	Compared YOLOv8, DenseNet, ResNet, and VGGNet, with YOLOv8 showing the best performance.	Resizing, Normalization, Data Augmentation (Rotation, Flipping, Translation)	YOLOv8 DenseNet ResNet VGGNet	[8]
Recognizing Wild Animals from Camera Trap Images Using Deep Learning	Demonstrates the effectiveness of DL, specifically VGG16, for detecting and classifying animals in wildlife conservation.	- Leveraged transfer learning with a pre-trained VGG16 model on ImageNet; - Loaded and preprocessed images to match VGG16 input size; - Decoded outputs for classification results, specifically for buffalo detection.	The VGG16 model was effective for detection and classification in conservation efforts.	Resizing, Normalization	VGG16	[9]

R-CNN Based Deep Learning Approach for Counting Animals in the Forest: A Survey	Presents a DL approach using R-CNN for classifying animals in forest camera trap images, relevant to rare species identification.	- Captured high-resolution images of forest environments; - Divided images into region proposals using a selective search algorithm; - Processed region proposals through a CNN to extract high-level features which were fed to SVMs for species classification; - Applied bounding box regression to refine localization.	The R-CNN approach was effective for classification in camera trap images.	Not specified	R-CNN CNN SVM	[10]
Multiclass Bird Species Identification using Deep Learning Techniques	DL models like EfficientNet-B0 can accurately identify bird species for ecosystem conservation.	- Employed four DL models (ResNet50, MobileNetV3, EfficientNet-B0, Wide-ResNet50V2) to identify 525 bird species; - Utilized transfer learning and fine-tuning to adapt pre-trained models; - Evaluated performance using precision, recall, and F1-score metrics.	EfficientNet-B0 performed best, but all models demonstrated high overall performance, with accuracy above 97%	Resizing, Normalization	ResNet50 MobileNetV3 EfficientNet-B0 Wide-ResNet50V2	[11]
Classification of Reptiles and Amphibians Using Transfer Learning and Deep Convolutional Neural Networks	A DL model using transfer learning and data augmentation classifies reptiles and amphibians with high accuracy, demonstrating potential for AI in biodiversity conservation.	- Used deep CNNs and Transfer Learning; - Optimized a pre-trained MobileNetV2 model on a large dataset of reptile and amphibian images; - Explored image augmentation techniques to improve model performance.	The MobileNetV2 model achieved high accuracy (82%); Data augmentation improved the model's ability to generalize across diverse conditions.	Contrast Optimization, Noise Reduction, Image Sizing, Data Augmentation	MobileNetV2	[12]

An Enhanced Animal Species Classification and Prediction Engine using CNN	A CNN-based approach achieves high accuracy for automated animal species classification, enabling more efficient wildlife conservation efforts.	- Used CNNs for animal species classification; - Transfer learning by fine-tuning pre-trained CNN models; - Employed a prediction engine and web application for user interaction.	The CNN approach achieved a high accuracy of 98%. Developed a web application for image input and species prediction.	Resizing, Normalization, Data Augmentation (Rotation, Flipping, Scaling)	CNNs	[13]
Automated Multi-Species Classification Using Wildlife Datasets Based on Deep Learning Algorithms	A DL model for automated multi-species classification of wildlife images achieves high accuracy and could be valuable for conservation applications.	- Used two CNN models, EfficientNetB0 and VGG16; - 37 distinct wildlife species; - Trained the model on a dataset of 185,111 images and tested it on 3,131 images;	Achieved over 80% accuracy and 90% top-5 accuracy in multi-species classification. EfficientNetB0 performed better.	Data cleaning to eliminate bias and irrelevant information Normalization Data Augmentation	EfficientNetB0, VGG16	[14]
Marine Animal Classification Using Deep Learning and Convolutional Neural Networks (CNN)	Demonstrates the effectiveness of CNNs for classifying marine animal species.	- Compressed and flattened the pixel values of the photos of marine animals; - CNN Model Architecture: convolutional, pooling, and fully connected layers (FCL); - Tuned the hyperparameters for optimal performance; - Deployed the trained model for performance evaluation and real-time forecasting.	The CNN model was effective for classifying marine animals.	Compressing and Flattening Pixel Values	CNNs (Pooling layers, FCL), EfficientNet B0/B3/B5, Grad-CAM (Gradient-weighted Class Activation Mapping)	[15]

Animal Species Recognition with Deep Convolutional Neural Networks from Ecological Camera Trap Images	DL models can classify rare animal species like snakes, lizards, and toads from camera trap images.	- Balanced the imbalanced dataset; - Investigated various image preprocessing techniques; - Applied data augmentation; - Trained and tested ML models to classify three animal groups (snakes, lizards, and toads) from camera trap images; - Experimented with pre-trained models (VGG16 and ResNet50) and a self-trained CNN.	The custom CNN with specific parameters performed best.	Image Preprocessing, Data Augmentation	VGG16 ResNet50 Custom CNN (CNN-1)	[16]
Automating bird species classification: A deep learning approach with CNNs	A CNN model using transfer learning and data augmentation achieved high accuracy in classifying 525 bird species.	- Dataset of 84,635 training images, 2,625 images both for validation and test; - Applied data augmentation; - Used callbacks like Model Checkpoint, EarlyStopping, Reduce LROnPlateau; - Used pre-trained EfficientNetB0 (transfer learning); - Trained for 150 epochs using Adam optimizer and categorical cross-entropy loss, with accuracy as the metric.	Achieved high accuracy in classifying 525 bird species.	Data Augmentation	EfficientNetB0	[17]
An Eagle-Eye Vision: Advancements in Avian Species Classification	CNNs are effective for automated bird species classification, with potential applications in avian conservation.	- Image Preprocessing (scaling, normalization, and data augmentation); - Feature Extraction using CNN architecture; - Transfer learning to fine-tune a pre-trained CNN model.	The CNN approach was effective for automated bird species classification.	Scaling, Normalization, Data Augmentation	CNNs	[18]

Assessing Convolutional Neural Network Animal Classification Models for Practical Applications in Wildlife Conservation	Assesses the performance of CNN models for identifying animal species in camera-trap images for wildlife conservation applications.	-Used the Wellington Camera Traps dataset; - Developed and analysed 10 different CNN models; - Tested the CNN models on 7 different datasets, simulating 13 possible project scenarios; - Evaluated model performance using top-1 and top-5 accuracy and false alarm rate and missed invasive rate to reflect wildlife conservation goals.	Different CNN models were evaluated, with performance reflecting conservation goals.	Not specified	CNNs	[19]
Fauna Image Classification using Convolutional Neural Network	A CNN model developed for classifying fauna images with high accuracy, aiding in wildlife conservation research.	- Trained and developed the CNN model to achieve high classification accuracy (91.84% training accuracy, 99.77% test accuracy); - Classified images of animals captured in dense forest environments.	The model achieved high accuracy in classifying fauna images.	Not specified	CNNs (VGG16)	[20]
Recognition Of Animal Species on Camera Trap Images Using Machine Learning and Deep Learning Models	DL models like InceptionV3 outperform machine learning algorithms in classifying animal species from camera trap images.	- Compared the performance of several ML algorithms (SVM, Random Forest) and DL models (AlexNet, Inception V3); - Used the KTH dataset with 19 different animal categories, selected 12 for evaluation;	DL models (AlexNet, Inception V3) outperformed machine learning (SVM, Random Forest) in terms of classification accuracy.	Not specified	SVM, Random Forest, AlexNet, Inception V3	[21]

Convolutional neural networks for cell detection and counting: a case study of human cell quantification in zebrafish xenografts using deep learning object detection techniques	This automated approach is applied to a specific, innovative research area at Fundação Champalimaud: quantifying cells in zebrafish xenografts used for studying cancer, metastasis, and drug discovery, contributing a practical application of object detection techniques to overcome common challenges like cell overlap, high cell density, and heterogeneity in medical imaging analysis.	- Preparing a labelled dataset of zebrafish cell images; - Selecting a powerful object detection model (Faster R-CNN with Inception ResNetV2); - Leveraging transfer learning from the COCO dataset; - Implementing and training the model using the TensorFlow Object Detection API; - Evaluating its cell counting performance using standard detection metrics like mAP.	The proposed Faster R-CNN model accurately detected and counted cells in challenging zebrafish xenograft images, achieving 89.5% mAP@0.50.	Data Augmentation (All the possibilities*[1])	Faster R-CNN, Inception ResNet V2, Transfer Learning	[22]

*[1] *HorizontalFlip, VerticalFlip, PixelValueScale, ImageScale, RGBtoGray, Adjustbrightness, Adjustcontrast, Adjusthue, Adjustsaturation, Distortcolor, Jitterboxes, Cropimage, Padimage, Croppadimage, Croptoaspectratio, Blackpatches, Rotation90*. Additionally, combinations were tested, such as applying only the top four performing augmentations (*HorizontalFlip, Adjustbrightness, Jitterboxes, Croptoaspectratio*), applying all listed techniques except *Adjusthue, Padimage*, and *Croppadimage*, and finally, applying all listed individual augmentations together.

APPENDIX B. EDA

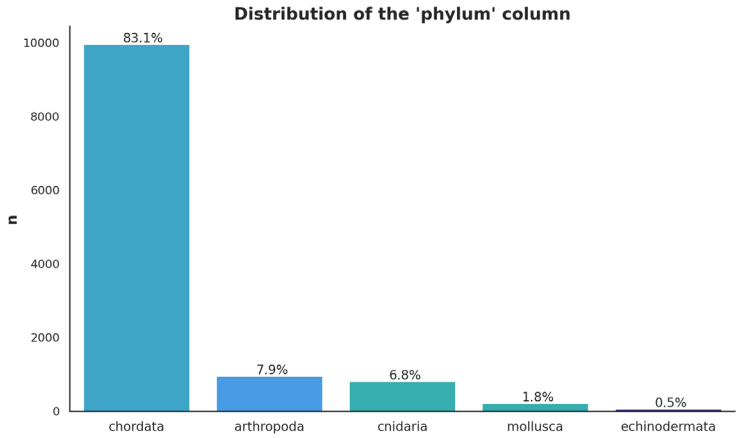


Figure B1 – Barplot for phylum distribution.

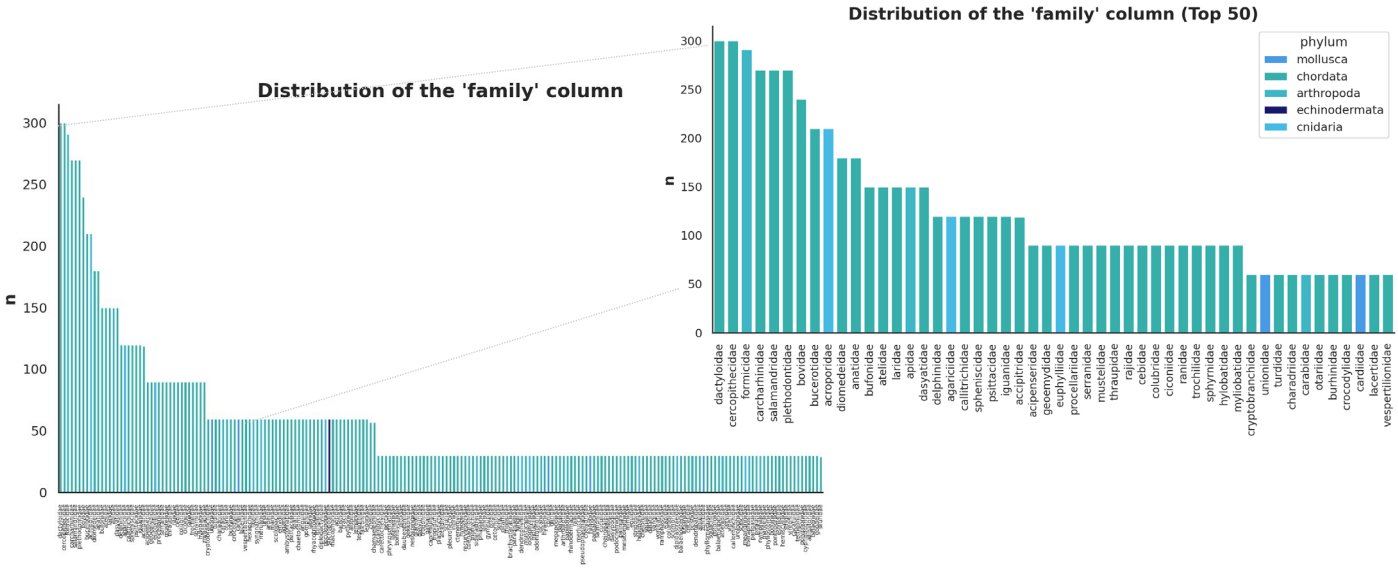


Figure B2 – Barplot for family distribution.

(**Left:** All families, **Right:** Top 50 families)

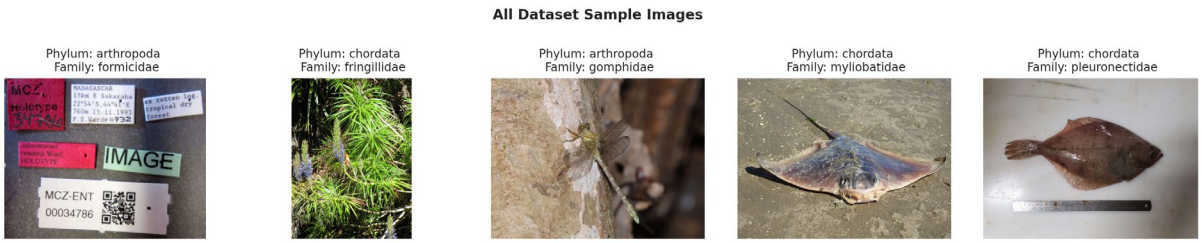


Figure B3 – Sample images from the dataset, illustrating diversity across phyla and potential non-animal images.

Figure B4 – Sample images from Training set .

(**Top to bottom:** crop_to_aspect_ratio=False & pad_to_aspect_ratio=False | crop_to_aspect_ratio=True | pad_to_aspect_ratio=True)

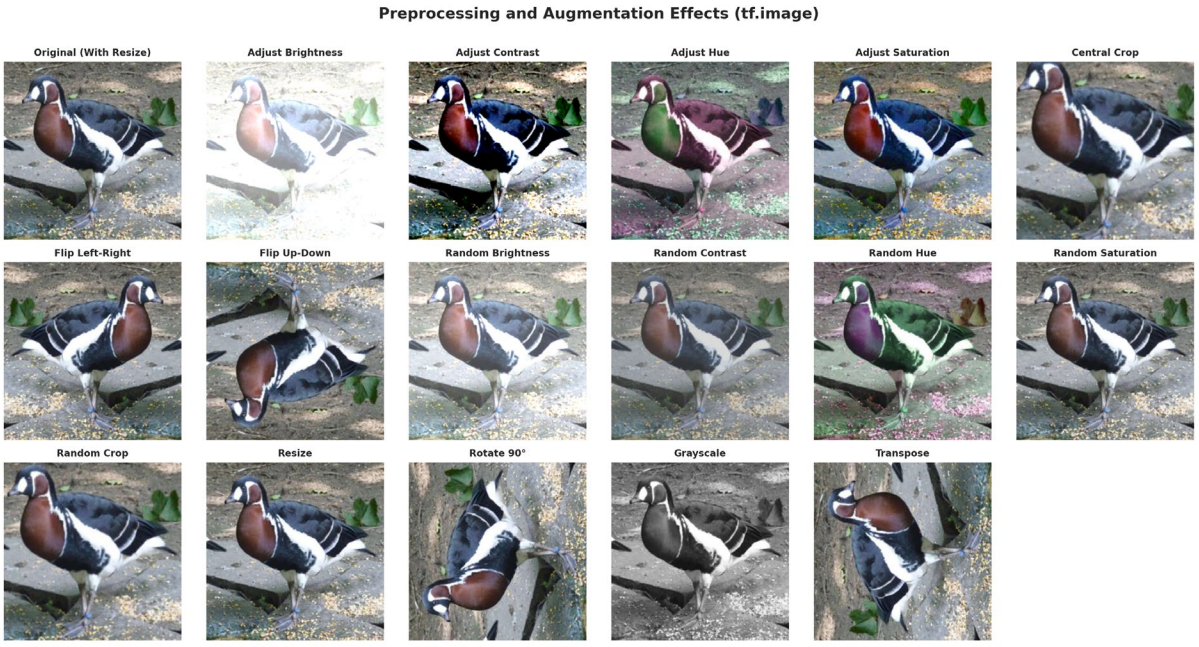


Figure B5 – Examples of potential *tf.image* preprocessing and augmentation effects.

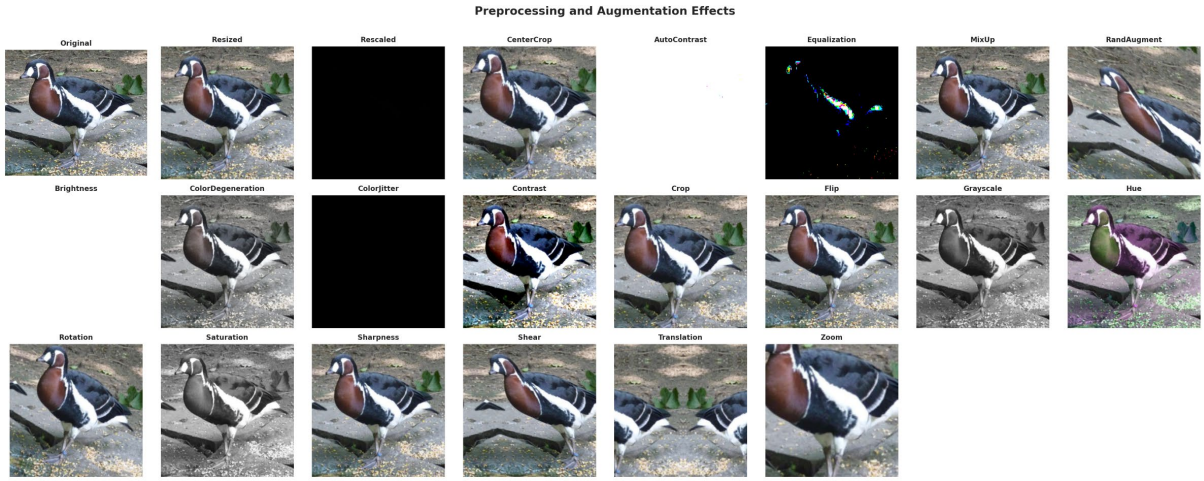


Figure B6 – Examples of image augmentations applied by *RandAugment*.

APPENDIX C. MODELLING

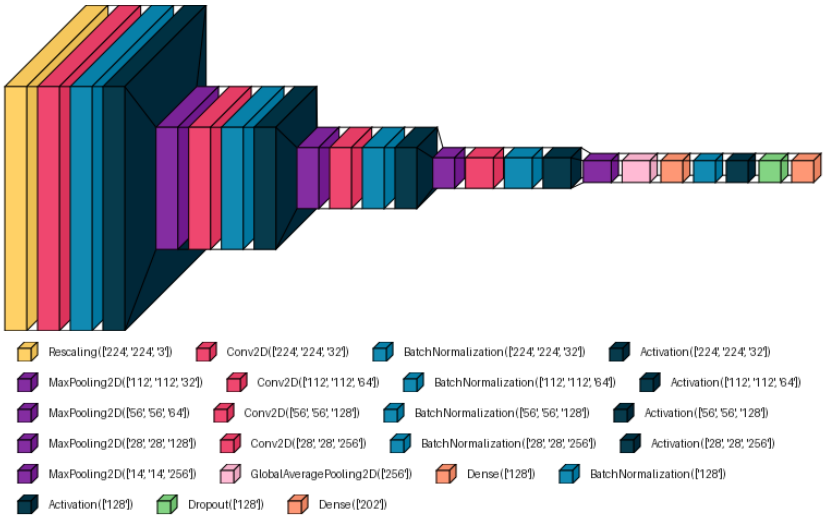


Figure C1 – Baseline CNN Model Architecture.

(**Note:** Visualization generated using the *visualkeras* library, showing all layers and their respective output dimensions.)

Table C1 - Baseline Model Combinations Selection Results (Max 10 Epochs).

Classification Models | Results																
	Training Set						Validation Set					Test Set				
Combination	Time of Execution	Accuracy	Precision	Recall	F1 Score	AUROC	Accuracy	Precision	Recall	F1 Score	AUROC	Accuracy	Precision	Recall	F1 Score	AUROC
Original | Grayscale=F | Contrast=F | Saturation=F	332,99	0,1487	0,7743	0,0258	0,0602	0,8334	0,1344	0,7333	0,0092	0,0453	0,8116	0,1309	0,8235	0,0117	0,0414	0,8023
Original | Grayscale=T | Contrast=F | Saturation=F	337,86	0,1090	0,6615	0,0090	0,0319	0,7922	0,0818	1,0000	0,0025	0,0153	0,7344	0,0834	0,5000	0,0008	0,0162	0,7363
Original | Grayscale=F | Contrast=T | Saturation=F	429,44	0,1516	0,7253	0,0245	0,0605	0,8433	0,1678	0,9167	0,0092	0,0573	0,8209	0,1518	0,8125	0,0108	0,0472	0,8188
Original | Grayscale=F | Contrast=F | Saturation=T	391,27	0,1393	0,7325	0,0240	0,0532	0,8332	0,1210	0,5000	0,0109	0,0421	0,7994	0,1284	0,7500	0,0175	0,0375	0,7941
SMOTE | Grayscale=F | Contrast=F | Saturation=F	1520,10	0,1400	0,7157	0,0176	0,1142	0,8403	0,1068	0,4634	0,0159	0,0895	0,7918	0,0942	0,3902	0,0133	0,0694	0,7790
SMOTE | Grayscale=T | Contrast=F | Saturation=F	1494,54	0,1167	0,7454	0,0116	0,0903	0,8157	0,0317	0,3750	0,0050	0,0286	0,6941	0,0442	0,2857	0,0050	0,0412	0,6943
SMOTE | Grayscale=F | Contrast=T | Saturation=F	1783,68	0,1508	0,7338	0,0208	0,1247	0,8472	0,1611	0,7708	0,0309	0,1184	0,8416	**0,1451**	0,6944	0,0209	**0,1042**	0,8348
SMOTE | Grayscale=F | Contrast=F | Saturation=T	2212,43	0,1426	0,7161	0,0178	0,1162	0,8409	0,1269	0,4783	0,0092	0,0888	0,7990	0,1226	0,6400	0,0133	0,0910	0,8051
Weights | Grayscale=F | Contrast=F | Saturation=F	577,22	0,0731	0,5333	0,0033	0,0513	0,7836	0,0843	1,0000	0,0058	0,0435	0,7432	0,0642	0,7500	0,0025	0,0335	0,7369
Weights | Grayscale=T | Contrast=F | Saturation=F	549,78	0,0576	0,4722	0,0018	0,0355	0,7470	0,0159	0,5000	0,0008	0,0094	0,6502	0,0175	0,0000	0,0000	0,0121	0,6375
Weights | Grayscale=F | Contrast=T | Saturation=F	595,86	0,0892	0,4819	0,0042	0,0612	0,8003	0,0843	0,9167	0,0092	0,0491	0,7752	0,0676	0,8750	0,0058	0,0448	0,7676
Weights | Grayscale=F | Contrast=F | Saturation=T	571,12	0,0836	0,5614	0,0033	0,0579	0,7911	0,0751	1,0000	0,0008	0,0562	0,7554	0,0717	0,0000	0,0000	0,0527	0,7497

Table C2 - Baseline & Pretrained Model Combinations Selection Results (Max 100 Epochs).

Classification Models | Results																	
		Training Set						Validation Set					Test Set				
	Combination	Time of Execution[*1]	Accuracy	Precision	Recall	F1 Score	AUROC	Accuracy	Precision	Recall	F1 Score	AUROC	Accuracy	Precision	Recall	F1 Score	AUROC
Baseline Model	Original	2051,11	0,262	0,828	0,074	0,171	0,915	0,232	0,958	0,058	0,144	0,860	0,236	0,816	0,059	0,135	0,864
	Original & Contrast	1100,64	0,185	0,771	0,040	0,089	0,868	0,136	0,875	0,012	0,063	0,803	0,130	0,667	0,007	0,072	0,797
	SMOTE	3515,03	0,165	0,750	0,028	0,140	0,860	0,149	0,660	0,029	0,123	0,829	0,143	0,642	0,028	0,116	0,832
	SMOTE & Contrast	3079,59	0,209	0,760	0,045	0,185	0,881	0,178	0,786	0,046	0,155	0,856	0,195	0,756	0,052	0,166	0,854
	SMOTE & Saturation	3837,12	0,206	0,782	0,046	0,182	0,881	0,166	0,705	0,052	0,130	0,837	0,154	0,650	0,053	0,122	0,846
VGG-19	Original	552,15	0,737	0,859	0,635	0,737	0,989	0,601	0,772	0,517	0,556	0,946	0,612	0,747	0,525	0,576	0,946
	Original & Contrast	508,68	0,711	0,832	0,623	0,706	0,985	0,603	0,763	0,514	0,557	0,947	0,598	0,741	0,531	0,564	0,942
	SMOTE	908,73	0,543	0,780	0,419	0,540	0,950	0,548	0,770	0,438	0,515	0,949	0,560	0,760	0,455	0,531	0,949
	SMOTE & Contrast	1063,21	0,542	0,768	0,422	0,540	0,946	0,555	0,763	0,452	0,526	0,948	0,548	0,734	0,445	0,527	0,946
	SMOTE & Saturation	1896,29	0,579	0,809	0,453	0,577	0,958	0,565	0,762	0,461	0,550	0,950	0,576	0,760	0,467	0,551	0,948
ResNet152V2	Original	374,39	0,844	0,924	0,759	0,843	0,998	0,629	0,798	0,575	0,585	0,953	0,621	0,755	0,550	0,566	0,949
	Original & Contrast	317,29	0,814	0,908	0,716	0,806	0,997	0,612	0,777	0,523	0,565	0,948	0,591	0,757	0,530	0,539	0,946
	SMOTE	1088,89	0,704	0,859	0,608	0,703	0,979	0,614	0,744	0,546	0,570	0,949	0,617	0,731	0,556	0,587	0,945
	SMOTE & Contrast	855,47	0,632	0,833	0,519	0,631	0,968	0,576	0,750	0,497	0,532	0,945	0,588	0,731	0,517	0,556	0,947
	SMOTE & Saturation	629,37	0,651	0,840	0,544	0,650	0,972	0,589	0,752	0,521	0,551	0,947	0,596	0,735	0,528	0,567	0,948
ConvNeXtBase	Original	3122,63	0,948	0,972	0,912	0,948	1,000	0,821	0,901	0,779	0,800	0,984	0,848	0,909	0,802	0,827	0,983
	Original & Contrast	3152,91	0,936	0,964	0,892	0,935	0,999	0,811	0,900	0,770	0,787	0,984	0,832	0,907	0,784	0,808	0,982
	SMOTE	11430,69	**0,886**	0,939	0,845	**0,886**	0,996	**0,820**	0,881	0,797	**0,804**	0,980	**0,846**	0,901	0,822	**0,829**	0,979
	SMOTE & Contrast	14101,30	0,866	0,931	0,817	0,866	0,995	0,824	0,880	0,786	0,799	0,980	0,831	0,884	0,810	0,816	0,981
	SMOTE & Saturation	7867,14	0,873	0,932	0,826	0,873	0,995	0,820	0,877	0,791	0,802	0,978	0,828	0,890	0,807	0,813	0,981
EfficientNetB0	Original	745,92	0,846	0,925	0,765	0,838	0,998	0,736	0,867	0,683	0,698	0,973	0,749	0,873	0,693	0,716	0,974
	Original & Contrast	562,20	0,800	0,904	0,691	0,792	0,995	0,701	0,854	0,617	0,652	0,967	0,736	0,871	0,650	0,706	0,969
	SMOTE	937,52	0,789	0,902	0,708	0,788	0,989	0,738	0,844	0,677	0,699	0,965	0,751	0,861	0,705	0,737	0,969
	SMOTE & Contrast	1096,19	0,707	0,879	0,595	0,706	0,980	0,699	0,838	0,620	0,665	0,963	0,719	0,852	0,651	0,693	0,966
	SMOTE & Saturation	581,61	0,751	0,887	0,656	0,750	0,985	0,719	0,853	0,653	0,690	0,971	0,736	0,850	0,665	0,727	0,973

[*1] **Note:** The execution times shown in the table are **not** meant to be compared across different models. They are only comparable **within** each model's set of configurations, as the code was run on hardware platforms with varying computational capabilities.

APPENDIX D. HYPER-PARAMETER GRIDSEACRH

Table D1 - Hyper-parameters used on *RandomSearch*.

Estimator for *RandomSearch*	**Hyper-parameter**	**Tested Parameters**
ConvNeXt - SMOTE	unfreeze_base	['True', 'False']
	learning_rate	[1e-3, 1e-4, 1e-5] 0.001 | 0.0001 | 0.00001
	optimizer	['adam', 'sgd', 'rmsprop']
	dropout_rate	[0.4, 0.5, 0.6, 0.7]

APPENDIX E. INNOVATIVE APPROACH

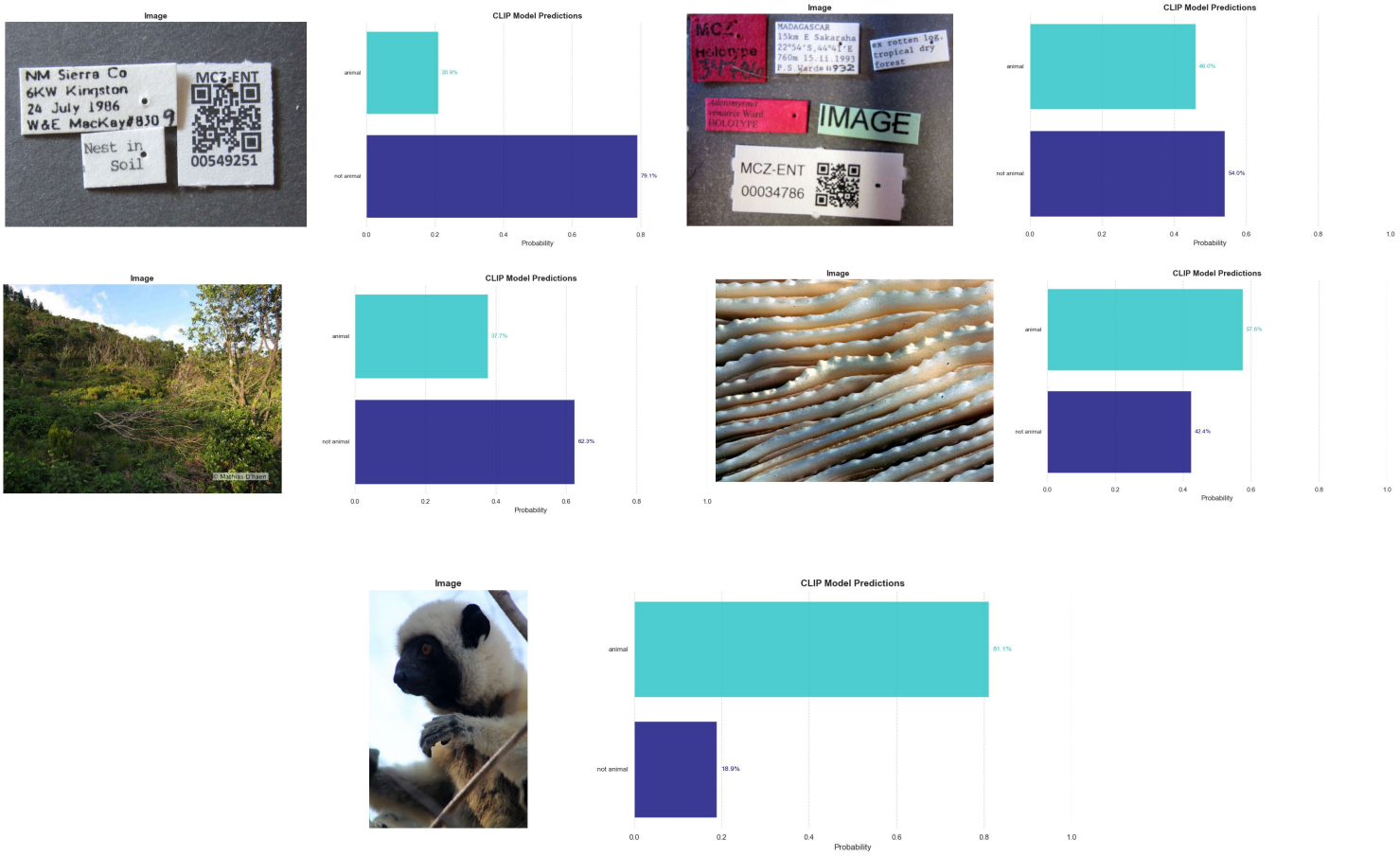


Figure E1 – Examples of classification using the CLIP model (*"animal"* vs. *"not animal"*).

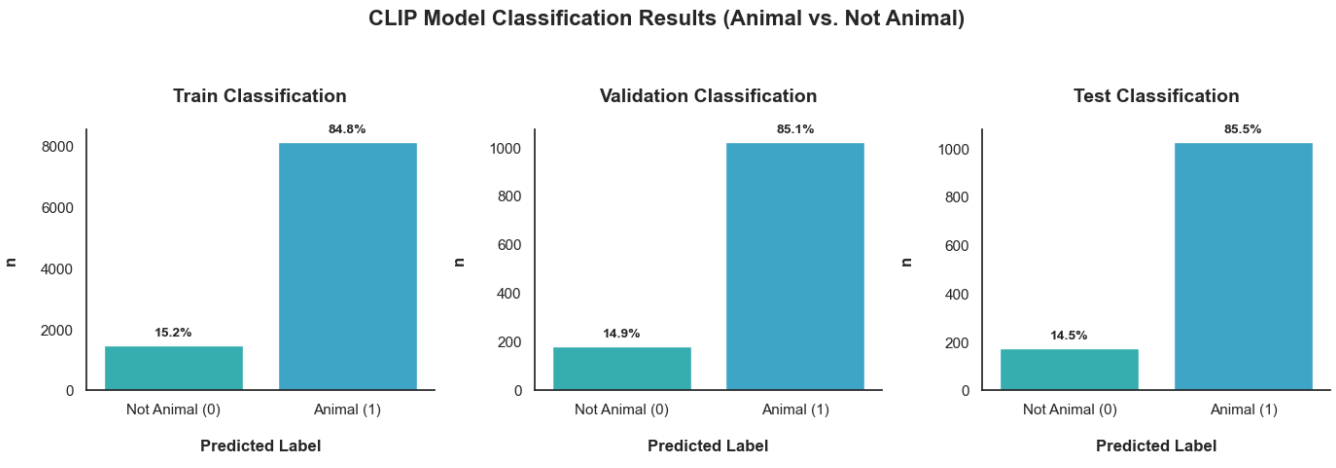


Figure E2 – Distribution of CLIP classification results (Animal vs. Not Animal) in train, validation, and test datasets.

APPENDIX F. BEST MODELS PREDICTIONS ANALYSIS

Table F1 – Final Model Performance Metrics after Retraining on CLIP-Filtered Data.

			Train					Validation					Test				
		Time of Execution	Accuracy	Precision	Recall	F1 Score	AUROC	Accuracy	Precision	Recall	F1 Score	AUROC	Accuracy	Precision	Recall	F1 Score	AUROC
Model	Variation																
ConvNeXtBase - Final Model	SMOTE (OnlyAnimals)	2451.89	0.8231	0.9198	0.7501	0.8224	0.991	0.8225	0.8959	0.7676	0.7761	0.9819	0.8312	0.8971	0.7912	0.7873	0.9855

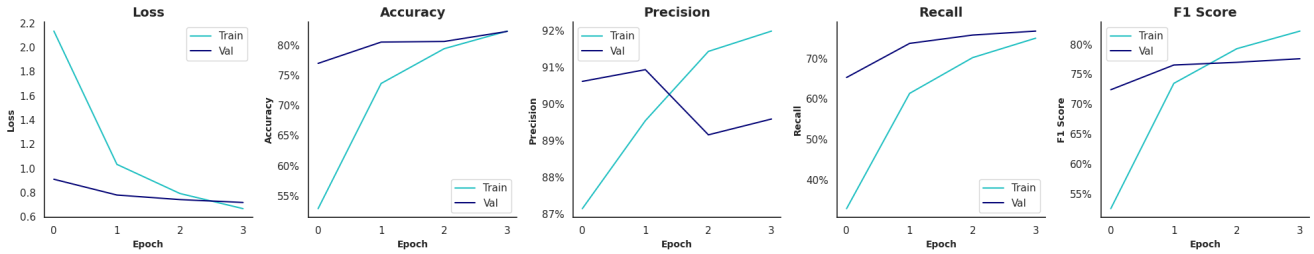


Figure F1 – Training and Validation Metrics (Loss, Accuracy, Precision, Recall, F1 Score) vs. Epoch for the Final Best Model.

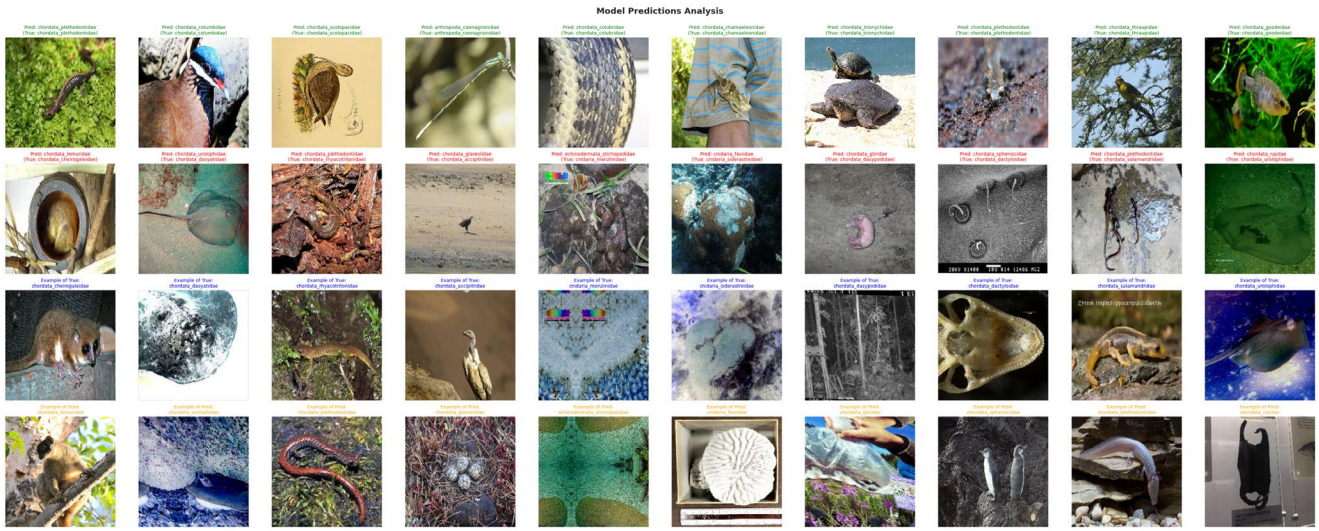


Figure F2 – Qualitative Prediction Analysis for the Final Best Model on the Test Set.

Correct Predictions (Row 1), Incorrect Predictions (Row 2), and Examples of True (Row 3) vs. Predicted (Row 4) classes for the errors.

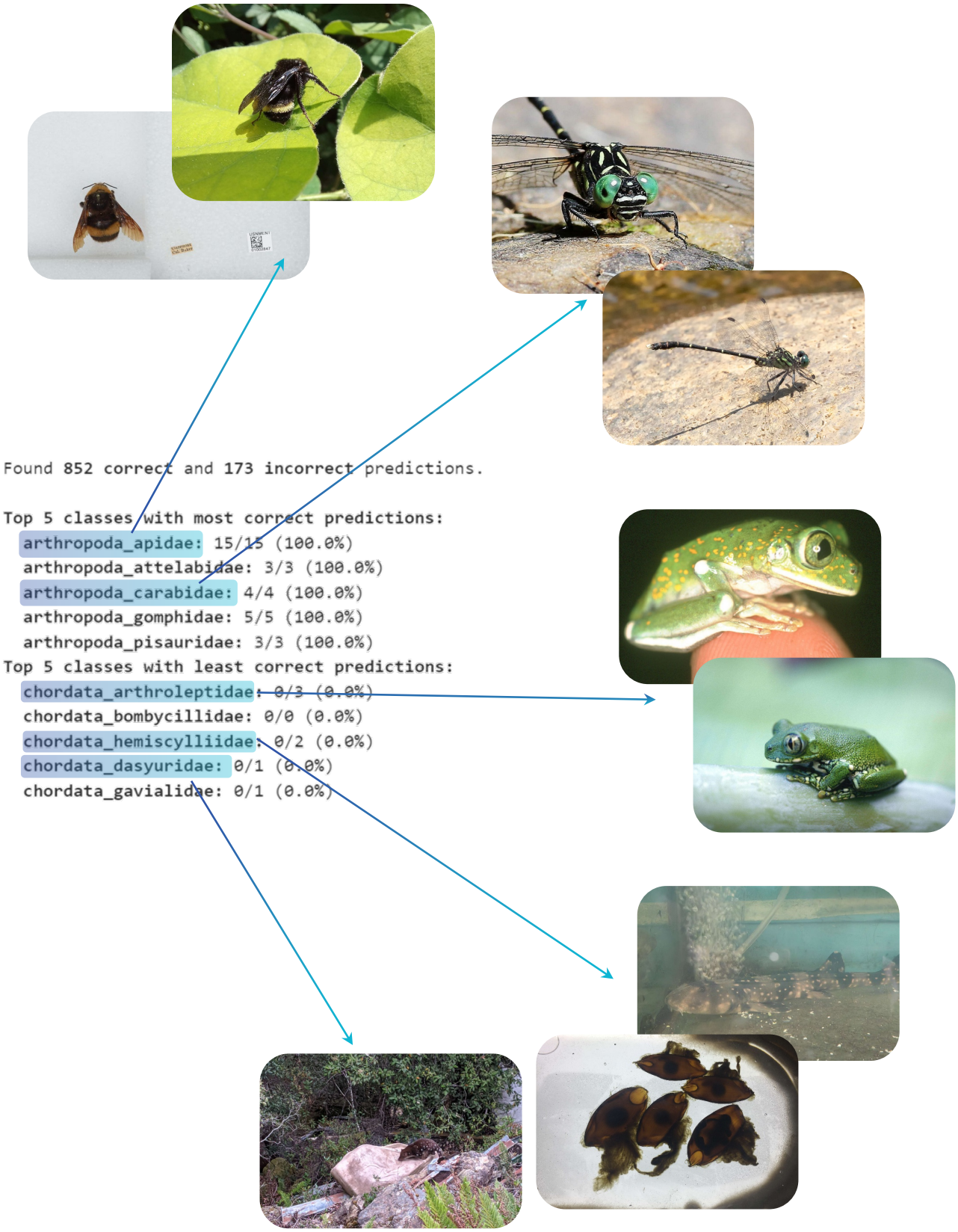


Figure F3 – Top 5 best-performing and Top 5 worst-performing families, with illustrative example images.

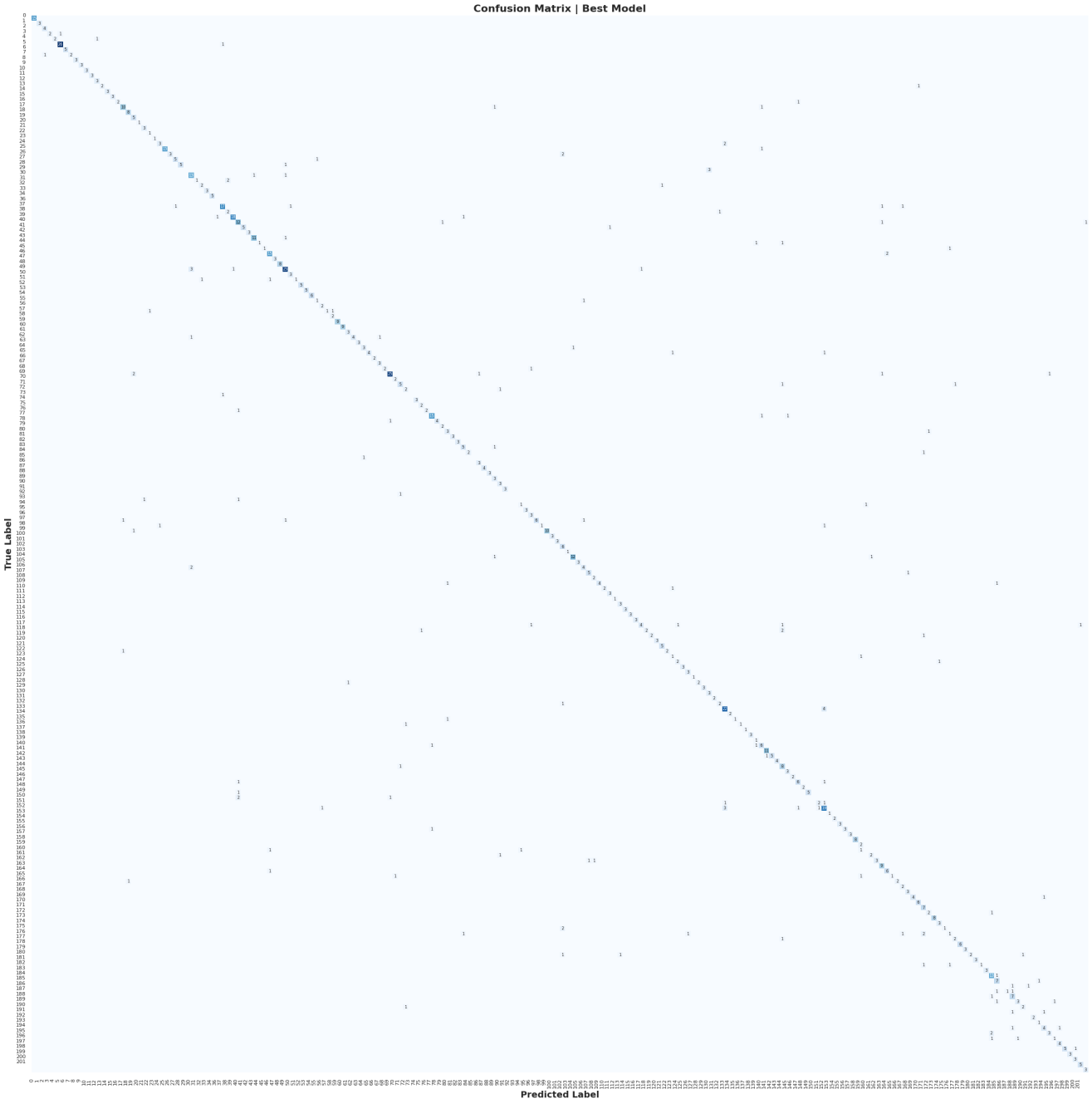


Figure F4 – Confusion Matrix for the Final Best Model on the Test Set.

ANNEX A. PRETRAINED MODELS

The following pre-trained models from Keras Applications [24], all using weights trained on ImageNet and expecting a (224, 224, 3) input size, were used in this project are described in **Table A1**.

Table A1 – Comparison of Pretrained CNN Architectures (*Keras Applications*).

Characteristic	VGG19 [27]	ResNet152V2 [28]	ConvNeXt Base [29]	EfficientNetV2 B0 [30]
Key Idea	Very deep stack of small (3×3) conv filters to increase depth.	Deep residual learning with "pre-activation" blocks to ease optimization of very deep nets.	Modern pure-ConvNet that "modernizes" ResNet toward Transformer design, using large kernels and simplified blocks.	Compound scaling of depth/width/input resolution, with training-aware NAS to jointly optimize accuracy and speed.
Architecture Highlights	16 Conv layers + 3 FC layers; uniform 3×3 kernels; 5 MaxPool layers.	152 layers of batch-norm → ReLU → conv ("pre-activation"); identity shortcuts.	Four stages with inverted bottleneck blocks, depth-wise convolutions, layer normalization, large 7×7 kernels; no attention.	MBConv blocks (inverted residuals + SE), progressive learning schedule; swish activation.
Input size & format	(224 × 224 × 3), BGR	(224 × 224 × 3), RGB	(224 × 224 × 3), RGB	(224 × 224 × 3), RGB
Depth (# layers)	19 weight layers (16 Conv. + 3 Fully Connected (FC)).	152 weight layers (bottleneck blocks)	≈ 101 layers (conv blocks + stage transitions)	≈ 82 layers (MBConv blocks + FC head)
Parameters [24]	≈ 20 million	≈ 60 million	≈ 90 million	≈ 6 million
ImageNet Top-1 Accuracy [24]	71.3%	78.0%	85.3%	77.1%
ImageNet Top-5 Accuracy [24]	90.0%	94.2%	-	93.3 %
preprocess_input	RGB→BGR, Zero-center each colour channel with respect to the ImageNet dataset (no scaling)	Scale to [−1, +1]	Scale to [0-255]	Scale to [0-255]
Reference	Simonyan & Zisserman (2015)	He et al. (2016)	Liu et al. (2022)	Tan & Le (2021)

ANNEX B. COMPARISON OF KERAS TUNER STRATEGIES

Hyperparameter tuning is essential for optimizing DL model performance [1]. *Keras Tuner* provides several built-in strategies (tuners) to automate this process. The choice of tuner involves trade-offs between computational cost, exhaustiveness, and the likelihood of finding optimal hyperparameters. Below is a comparison of three common *Keras Tuner* strategies: *GridSearch*, *RandomSearch*, and *Hyperband*.

Table B1 – Comparison of Keras Tuner Search Strategies.

Feature	GridSearch [1]	RandomSearch [35]	Hyperband [36]
Search Strategy	Exhaustively tests every combination defined in a discrete grid	Randomly samples parameter combinations from specified distributions or lists.	Adaptive resource allocation using successive halving; prioritizes promising configurations early.
Efficiency	Very low; computationally expensive, scales exponentially with the number of parameters ("curse of dimensionality".	High; significantly more efficient than Grid Search for a given computational budget.	Very high; typically, faster than Random Search, especially for models where early performance is indicative.
Exhaustiveness	Guarantees finding the best parameters within the defined grid.	Not guaranteed to find the global optimum but often finds very good solutions quickly.	Not guaranteed to find the global optimum; may prune configurations that improve slowly.
Pros	- Simple concept. - Guaranteed to find best grid point. - Reproducible.	- Computationally efficient. - Effective even with high-dimensional spaces. - Often outperforms Grid Search.	- Highly resource-efficient due to early stopping. - Excellent for expensive model training (e.g., Deep Learning). - Balances exploration/exploitation.
Cons	- Impractical for large search spaces. - Only tests predefined discrete values. - May miss optimal values between grid points.	- Performance depends on the number of trials and randomness. - Might not finely explore the most promising regions.	- More parameters to configure (**max_epochs**, **factor**). - Assumes early performance correlates with final performance. - Can discard "late bloomers".
Use Case	Very small search spaces with few parameters where exhaustiveness is critical.	Good general-purpose tuner, often a strong baseline. Suitable for large or continuous search spaces.	Preferred for computationally expensive models (DL) where training time is a major constraint and early stopping is effective.

Conclusion

The selection of a hyperparameter tuning strategy in **Keras Tuner** relies on the exact constraints and goals of the project. ***GridSearch*** offers exhaustiveness over a defined grid but is often computationally infeasible for complex models or large search spaces. ***RandomSearch*** provides a significant improvement in efficiency by randomly sampling the search space and frequently yields better results than *GridSearch* within the same time budget, making it a robust default choice. ***Hyperband*** represents a more advanced, adaptive strategy that optimizes resource allocation through early stopping of less promising trials. It is particularly well-suited for tuning computationally expensive deep learning models where training time is a bottleneck, provided that early performance is a reasonable indicator of final performance. *Keras Tuner* simplifies the implementation of all three, allowing practitioners to choose the most appropriate trade-off between computational resources and the thoroughness of the hyperparameter search.