Camera Trap Wildlife Image Classification Using Deep Learning

Helin Melisa Ergezen
Master in Computer Vision
Porto University
Porto, Portugal
up202502605@edu.fe.up.pt

Gagandeep Kaur
Master in Computer Vision
University of Santiago de Compostela
Santiago, Spain
gagandeepkaur.gagandeep@rai.usc.es

Carolina Gomes
Master in Computer Vision
Porto University
Porto, Portugal
up202500454@edu.fe.up.pt

Eda Ozge Ozler
Master in Computer Vision
University of Santiago de Compostela
Santiago, Spain
edaozge.ozler@rai.usc.es

***Abstract*—Camera traps generate massive image datasets for wildlife monitoring, creating a need for automated species classification. We present a baseline pipeline using a frozen ResNet50 backbone with a lightweight classification head, site-aware train/validation splitting, class weighting, and data augmentation to handle real-world variability. On 16,488 images from *Taï National Park*, the model achieves early validation convergence, accurately recognizing prominent species while struggling with rare or subtle classes. This approach provides a reproducible baseline for scalable wildlife monitoring and future model improvements.**

I. Introduction

Monitoring wildlife in their natural habitats is critical for conservation efforts, but it can be challenging due to the sheer volume of data collected. Camera traps (automated surveillance systems triggered by motion or heat) allow researchers to observe animals without disturbing their behavior. These devices generate massive numbers of images, making manual analysis time-consuming and often impractical.

This project focuses on automating wildlife species classification from camera trap images using machine learning. Specifically, we aim to identify seven types of animals, as well as images containing no animals, from a dataset collected in *Taï National Park* by the Wild Chimpanzee Foundation and the Max Planck Institute for Evolutionary Anthropology.

By leveraging computer vision techniques, this project seeks to efficiently process camera trap data, enabling conservationists to track species, quantify observations, and make informed decisions to protect wildlife. The challenge also highlights the importance of building models that generalize well to new locations, as training and testing sites do not overlap.

This work is part of a DrivenData competition, providing a practical, hands-on opportunity to explore image classification in the context of wildlife conservation while contributing to real-world ecological research [1].

II. Exploratory Data Analysis

To gain insights into the dataset and guide model development, an exploratory analysis of the camera trap images was performed. The training set consists of 16,488 images across 148 sites, while the test set includes 4,464 images from 51 distinct sites, with no overlap between them. This setup ensures that models must generalize to unseen locations, a critical consideration for real-world conservation applications.

The dataset captures eight categories: seven wildlife species (antelope duiker, bird, civet genet, hog, leopard, monkey prosimian, rodent) and blank images containing no animals. Class distribution is moderately imbalanced, with monkey prosimian as the most frequent class (15.1%) and hog the least frequent (5.9%), yielding an imbalance ratio of 2.55x. This suggests that strategies such as class weighting or data augmentation may be necessary.

Fig. 1. Class distribution within the training dataset.

Analysis of site-level distribution reveals substantial variation in the number of images per site, with some sites contributing far more images than others. A sample of 100 images shows diverse resolutions (160–960 px width, 120–540 px height), aspect ratios (1.33–1.91), and file sizes (1.2–97.9 KB), reflecting real-world variability in camera trap data.

Finally, an overview of representative images for each class (Figure 2) illustrates the variability and ambiguity present in the dataset. Differences in animal size, orientation, partial occlusion, background complexity, and lighting are apparent, and blank images show natural vegetation or empty scenes that may resemble animal presence. This visual inspection highlights the challenges for automated classification and motivates the use of robust preprocessing, data augmentation, and feature extraction strategies to improve model performance across all classes.

Fig. 2. Representative images from each class showing variability in appearance, background, and potential ambiguity in classification.

III. Data Preparation and Feature Engineering

A. Dataset Overview and Site-Based Splitting

The data is provided in two CSV files containing image metadata and multi-label annotations. To prevent information leakage, a site-based splitting strategy is employed where entire camera trap sites are assigned exclusively to either training or validation sets, as images from the same site share environmental characteristics that could enable the model to memorize site-specific features rather than learning generalizable animal classification patterns.

The splitting process uses stratified sampling based on each site's dominant animal class to ensure balanced ecological representation. Using scikit-learn's `train_test_split` with a 20% validation ratio results in 118 training sites (13,518 images) and 30 validation sites (2,970 images). This approach achieves zero site overlap while maintaining proportional class distributions across both splits.

B. Class Imbalance and Weighting

Analysis reveals substantial class imbalance, with civet_genet (2,128 samples) appearing 2.4 times more frequently than hog (891 samples) in the training set. To mitigate this imbalance, class-specific weights are computed using the inverse frequency formula:

$$w_c = \frac{n_{total}}{n_{classes} \times n_c} \tag{1}$$

where w_c is the weight for class c, n_{total} is 13,518 total training images, $n_{classes}$ is 8, and n_c is the number of positive samples for that class. The computed weights range from 0.794 (civet_genet) to 1.896 (hog), ensuring balanced learning signals during training.

Implementation note: Although the task is single-label multi-class, we implemented a weighted BCE-with-logits loss on one-hot targets to conveniently incorporate per-class weights. During validation and inference, logits were converted to a proper multiclass probability distribution via softmax, and metrics were computed using multiclass log loss and top-1 accuracy.

C. Feature Engineering and Validation

Feature engineering occurs primarily through data augmentation rather than manual feature extraction. Images are normalized using ImageNet statistics to enable transfer learning from pre-trained models. The site metadata informs the splitting strategy but is not used as a direct model input. Comprehensive validation confirms zero site overlap, presence of all classes in both splits, and appropriate validation ratio (18.0%).

IV. Dataset Construction and Data Loading

A. Custom Dataset Implementation

A custom `WildlifeDataset` class is implemented by inheriting from `torch.utils.data.Dataset`. The class accepts a pandas DataFrame with image metadata and labels, a data directory path, an Albumentations transform pipeline, and a test flag. The `__getitem__` method loads images using PIL, applies transformations, and returns a dictionary containing the image tensor (shape: [3, 192, 192]), multi-label vector (shape: [8]), image ID, and site metadata. Error handling returns a blank image if loading fails, preventing training interruptions.

B. Image Preprocessing and Augmentation

All images are resized to 192×192 pixels and normalized using ImageNet statistics (mean: [0.485, 0.456, 0.406], std: [0.229, 0.224, 0.225]) for transfer learning compatibility. The training augmentation pipeline includes horizontal flipping (50% probability) and color jittering (30% probability, ±20% brightness/contrast/saturation, ±5% hue) to simulate varying lighting conditions across camera trap sites. The validation pipeline applies only resizing and normalization without stochastic augmentation to ensure reproducible evaluation.

C. DataLoader Configuration

PyTorch DataLoaders are configured with batch size 64, resulting in 211 training batches and 47 validation batches per epoch. The training loader enables shuffling, pin memory for faster GPU transfer, and drops the last incomplete batch

for consistent batch sizes. The validation loader disables shuffling for deterministic evaluation. Multi-process loading (num_workers) is set to 0 due to environment constraints. Pipeline validation confirms correct tensor shapes, normalized pixel ranges, and proper multi-label format.

V. Model Training and Validation

A. Experimental Setup and Evaluation Protocol

Following the site-based split, we trained a baseline deep classifier under CPU-only constraints (PyTorch 2.8.0, CUDA unavailable). As described in the data-loading section, the pipeline supports different batch sizes; however, due to CPU memory and runtime stability, the final run reported here used batch size 32, resulting in 422 training batches and 93 validation batches per epoch. All images were resized as we mentioned earlier in the Image Processing part to ensure compatibility with ImageNet-pretrained backbones.

We evaluated the model after every epoch on the validation split using (i) top-1 accuracy and (ii) multiclass log loss, which is also the competition's primary objective. Log loss was computed from softmax probabilities using the true class index:

$$\mathcal{L}_{LL} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(p_{i,y_i}\right), \tag{2}$$

where p_{i,y_i} is the predicted probability assigned to the correct class. Compared to accuracy, log loss penalizes poorly calibrated predictions and confident misclassifications more strongly, which is particularly relevant in camera-trap settings where rare classes may be systematically under-confident.

To prevent overfitting, we applied early stopping with patience 6 based on validation log loss. In addition, training progress (train/val loss, train/val accuracy, validation log loss, and learning rate) was recorded using TensorBoard. Model checkpoints were saved whenever validation log loss improved, ensuring that the best-generalizing model was retained even if later epochs overfit.

B. Model Architecture and Transfer Learning Strategy

We adopted ResNet50 initialized with ImageNet weights and trained in a feature-extraction regime by freezing the full backbone and optimizing only the classification head. This choice is coherent with the dataset design: images from the same site share background and illumination patterns, and a fully fine-tuned backbone can unintentionally learn site-specific correlations rather than species cues. Freezing substantially limits this failure mode while also reducing compute cost on CPU.

With the backbone frozen, trainable parameters were reduced to 16,392 out of 23,524,424 (99.93% reduction). The original ResNet50 classifier was replaced by a compact head: Dropout ($p = 0.5$) followed by a fully-connected layer mapping the 2048-dimensional pooled feature vector to 8 logits. Dropout improves robustness by discouraging reliance on a narrow set of activations in the head, which is important given class imbalance and cluttered backgrounds.

C. Optimization, Scheduling, and Regularization

Only the classification head was optimized using AdamW with learning rate 10^{-3} and weight decay 10^{-4}. To adapt the step size based on generalization performance, we used ReduceLROnPlateau driven by validation log loss; if log loss did not improve for 3 consecutive epochs, the learning rate was multiplied by 0.5. In the reported run, the learning rate dropped to 5×10^{-4} at epoch 7 after a sustained plateau in validation log loss. This schedule enables faster learning early in training and smaller, more stable updates once the optimization approaches a local minimum.

Regularization in this baseline is multi-layered: (i) freezing the backbone (capacity control), (ii) Dropout in the head, (iii) weight decay in AdamW, (iv) data augmentation (Section V-E), and (v) early stopping. This combination specifically targets overfitting to site appearance and background textures, which is a primary risk in camera-trap imagery.

D. Data Augmentation and Preprocessing

To improve invariance to nuisance factors identified in EDA (illumination variation, slight viewpoint changes, partial occlusion, and sensor noise), we applied the following training-time augmentations: horizontal flip ($p = 0.5$), random brightness/contrast adjustments ($\pm20\%$), additive Gaussian noise ($\sigma = 0.01$), and small rotations ($\pm10°$). Validation preprocessing remained deterministic (resize + normalization only) to ensure that validation metrics reflect true generalization rather than stochastic augmentation effects.

E. Training Results, Checkpointing, and Model Selection

Training was configured for up to 12 epochs, but early stopping terminated the run at epoch 8. Validation log loss improved rapidly during the first two epochs and reached its minimum at epoch 2 (1.6188). After epoch 2, training loss continued decreasing while validation log loss increased, indicating overfitting starting around epoch 3. This pattern is consistent with the dataset characteristics: limited examples per rare class and strong site-specific correlations.

We checkpointed the model whenever validation log loss improved and selected the final inference model using:

$$\theta^* = \arg\min_t \mathcal{L}^{(t)}_{LL,\text{val}}. \tag{3}$$

VI. Inference and Prediction

A. Inference Pipeline

Inference used the best checkpoint (epoch 2) and was applied to 4,464 unlabeled test images from unseen sites, requiring robust generalization to new environmental conditions. The inference pipeline was intentionally matched to validation preprocessing to avoid distribution shift: each image was converted to RGB, resized with bilinear interpolation to 192×192, and normalized with ImageNet statistics. The model was run in evaluation mode, disabling Dropout and using fixed BatchNorm running statistics, and predictions were generated under `torch.no_grad()` to reduce memory usage.

TABLE I
TRAINING SUMMARY (CHECKPOINT USED FOR INFERENCE).

Item	Value
Train / Val images	13,518 / 2,970
Sites (train / val)	118 / 30
Classes	8
Backbone / head	ResNet50 (frozen) / Dropout(0.5)+FC
Total / trainable params	23,524,424 / 16,392
Input size / batch size	192×192 / 32
Optimizer	AdamW (lr 10^{-3}, wd 10^{-4})
Scheduler	ReduceLROnPlateau (factor 0.5, patience 3)
Early stopping	patience 6 (val log loss)
Best epoch	2
Best val log loss	1.6188
Best val accuracy	0.4017
Runtime / epochs executed	190.8 min / 8

Class probabilities were computed using softmax:

$$p_{ic} = \frac{\exp(z_{ic})}{\sum_{k=1}^{8} \exp(z_{ik})}. \tag{4}$$

Inference used batch size 256 (18 batches). On CPU, this required approximately 15–20 minutes, corresponding to roughly 223–298 images/min depending on system load. Larger batches were feasible at inference because gradients are not stored and no backward pass is executed.

B. Numerical Post-processing and Submission Validation

To ensure numerical stability under log-loss scoring, predicted probabilities were clipped to $[10^{-7}, 1-10^{-7}]$. Because clipping can slightly violate the probability simplex constraint, we renormalized each row to enforce $\sum_{c=1}^{8} p_{ic} = 1$. This step prevents undefined values in $\log(\cdot)$ computations and ensures the submission strictly satisfies competition requirements.

The submission file was validated for: (i) correct column ordering (`id` followed by the 8 class probability columns in the required order), (ii) correct row count (4,464), (iii) absence of missing values, (iv) probabilities within $[0, 1]$, and (v) per-row probability sums equal to 1 within tolerance. All checks passed. The final CSV size was 440.1 KB.

C. Optional Test-Time Augmentation

We implemented test-time augmentation (TTA) but did not use it in the final submission due to CPU overhead. The proposed configuration averages predictions over three views (original image, horizontal flip, and brightness/contrast adjustment). TTA can improve robustness and probability calibration, which directly impacts log loss; however, it increases inference time by approximately $3\times$, and was therefore omitted for this baseline.

VII. RESULTS AND DISCUSSION

This section analyzes the performance of the proposed ResNet50-based wildlife image classification system, focusing on training dynamics, generalization behavior, and prediction characteristics on the unseen test set.

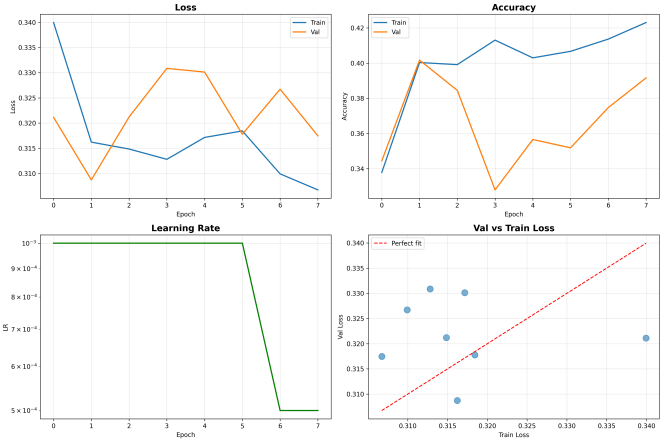

Fig. 3. Training and validation performance of the ResNet50 baseline model. Validation log loss reaches its minimum at epoch 2, after which overfitting is observed, motivating early stopping and selection of the epoch-2 checkpoint for inference.

A. Training Dynamics and Model Selection

The evolution of training and validation performance is shown in Fig. 3. The model demonstrates rapid initial learning, with validation log loss decreasing sharply during the first two epochs. The best validation log loss of 1.6188 is achieved at epoch 2, after which validation performance degrades while training loss continues to decrease. This divergence indicates the onset of overfitting.

Early stopping based on validation log loss successfully prevents further degradation of generalization performance. Consequently, the checkpoint from epoch 2 is selected for all subsequent inference experiments. These results confirm that, given the dataset size and class imbalance, freezing the backbone and training a lightweight classification head is an effective baseline strategy.

The epoch-2 checkpoint (validation loss 0.3087, validation accuracy 0.4017) was therefore used for all test-time inference. Table I summarizes the training configuration and outcomes.

B. Test-Set Prediction Characteristics

Fig. 4 illustrates the distribution of top-1 predicted classes across the test set. Predictions are dominated by antelope_duiker (30.4%), monkey_prosimian (28.4%), and leopard (18.0%), while bird (0.3%) and blank (1.5%) are rarely selected.

This behavior is consistent with camera-trap challenges observed in the EDA: (i) visually prominent mammals (large body size, distinctive textures) are easier to classify, (ii) birds often occupy a small region of the image and show high intra-class variability, and (iii) "blank" requires learning the absence of an animal, which can be confused with vegetation motion, shadows, or high-frequency background textures. Although class weighting increases minority-class gradient contributions during training, these classes remain difficult under a frozen-backbone baseline.

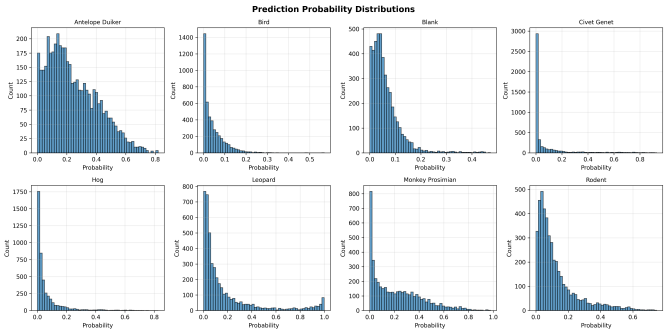

Fig. 4. Distribution of top-1 predicted classes on the test set. Predictions are concentrated in visually dominant species, while rare or subtle classes such as bird and blank are infrequently selected, highlighting remaining challenges due to class imbalance and visual ambiguity.

C. Discussion

Overall, the baseline model demonstrates reasonable discriminative capability given the constraints of CPU-only training and limited labeled data. The early saturation of validation performance suggests that further gains will likely require either partial fine-tuning of deeper backbone layers, stronger regularization, or the incorporation of test-time augmentation. Nevertheless, the current approach establishes a solid and reproducible baseline aligned with the goals of wildlife monitoring and ecological analysis.

VIII. Conclusion and Future Work

In this work, we presented a complete baseline pipeline for camera-trap wildlife image classification using a frozen ResNet50 backbone and a lightweight classification head. The proposed system incorporates site-aware data splitting, class imbalance handling via inverse-frequency weighting, and robust data augmentation. Experimental results demonstrate that the model achieves its best generalization performance early in training, with early stopping playing a crucial role in preventing overfitting.

Analysis of test-set predictions reveals strong performance on visually prominent species, while rare and subtle classes remain challenging. These findings are consistent with known limitations in wildlife image datasets, where environmental noise, occlusion, and severe class imbalance are common.

Future work will focus on improving generalization through partial backbone fine-tuning, stronger augmentation strategies, and the integration of test-time augmentation to enhance probability calibration. Additionally, incorporating temporal context from camera trap sequences and leveraging self-supervised or semi-supervised pretraining may further improve recognition of underrepresented species. These extensions offer promising directions toward more reliable and scalable wildlife monitoring systems.

References

[1] DrivenData. (n.d.). Conser-vision practice area: Image classification for wildlife conservation. Available: https://www.drivendata.org/competitions/87/competition-image-classification-wildlife-conservation/page/483/