The Effect of Data Balancing Techniques in RMSProp and Adam Optimizers in Deep Learning Models for Wildlife Recognition: A Pathway to Image-Based Visual Servoing (IBVS)

Iman Herwidiana Kartowisastro [1,2,*], Cuk Tho [3], and Reina Setiawan [3]

[1] Department of Computer Science, BINUS Graduate Program-Doctor of Computer Science, Bina Nusantara University, Jakarta, Indonesia
[2] Computer Engineering Department, Faculty of Engineering, Bina Nusantara University, Jakarta, Indonesia
[3] Department of Computer Science, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia
Email: ihkartowisastro@binus.ac.id (I.H.K.); cuktho@binus.ac.id (C.T.); reina@binus.ac.id (R.S.)
*Corresponding author

***Abstract*—Data collection in animal ecology has become more common, and extracting information can be done using deep learning methods. Challenges arise when using the dataset as the training input, including balancing the data quality and quantity. This research aims to investigate the effect of data balancing techniques on the optimizer—specifically, Root Mean Square Propagation (RMSProp) and Adam—in a deep learning image recognition model. Once recognition capability is obtained, it provides feedback for an Image-Based Visual Servoing (IBVS) system. The wildlife dataset is from Serengeti National Park and contains 41,762 images across 7 classes. Before applying data-balancing techniques, we determined that 1,100 images were needed, which we considered a balanced distribution across all classes since it was close to the original dataset. The next step was applying the balancing technique. We offered two balancing techniques: the first was to augment three classes with each class's 1,100 images, while the other four classes retained their original number of images. The second involved reducing the number of larger classes to 1,100 images. The result showed that combining the deep learning model AlexNet with RMSProp achieved the highest Macro F1-Score of 0.8118, outperforming the other deep learning models with the Adam optimizer. The best balancing technique reduced the larger class to a fixed number of images in the smaller dataset, and AlexNet with RMSProp is a good choice for real-time visual servoing.**

***Keywords*—data balancing techniques, wildlife animal recognition, image-based visual servoing, Root Mean Square Propagation (RMSProp), Adam, deep learning**

I. Introduction

In recent years, sensor technology has improved, making data collection more common in animal ecology. However, there are still problems when it comes to turning that data into useful and valuable information. This makes it harder to extract additional benefits or make the most of the vast dataset those sensors collect, such as by applying machine learning techniques with sufficient domain knowledge. The declining trend in animal diversity affects more than just genetics; it also impacts ecological and behavioral diversity [1].

Deep learning can be a solution for identification, especially when working with large datasets. Debauche *et al.* [2] showed that machine learning techniques, including deep learning, can detect animal behaviors such as feeding, movement, and illness in farm animals. The study concluded that Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), as part of deep learning, performed better than traditional video-based methods because they can learn both spatial and temporal features.

CNNs, as deep learning methods, play an important role in object recognition tasks. The layered structure helps extract features from complex visual inputs, making it well-suited to analyzing wildlife images from diverse and challenging environments. The development of CNN Architectures, from AlexNet in 2012 to VGG, GoogleNet, ResNet, SENet, and the most recent MobileNet has contributed greatly to the field's progress [3]. In addition, hyperparameters in CNN architectures significantly impact final classification performance.

Schindler and Steinhage [4] used CNNs to detect, categorize, and evaluate animal behavior from video recordings. Each video taken with an infrared camera and a flash had a frame rate of 8 frames per second. CNNs were also employed by Gullapelly and Banik [5] to more easily distinguish rigid (e.g., cars) from non-rigid (e.g., people) objects in video surveillance. On a dataset of 2,000 images, consisting of 1,000 rigid and 1,000 non-rigid images, the CNN achieved 91% accuracy, which was better than KNN

Manuscript received October 31, 2025; revised December 4, 2025; accepted February 3, 2026; published June 10, 2026.

and HaarCascade. This also shows that CNNs are robust enough to work with low-quality images.

The optimization function plays an important role in a CNN, improving learning by measuring how closely the model's predictions match the actual target. Melinte and Vladareanu [6] conducted a study to enhance human-robot interaction through facial expression recognition. This study used four optimizers, including Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSProp), Adam, and RAdam, for the CNN. Adam and RAdam outperform SGD and RMSProp, achieving accuracies of 92.4% and 94.1%, respectively. Adam outperforms SGD and RMSProp because of its adaptive learning rate and built-in momentum. RAdam addresses instability in Adam's early training phase via variance rectification.

Automated wildlife species recognition from images—originating from camera traps, drones, or remote sensing—is increasingly critical for ecological monitoring, species conservation, and biodiversity studies. A challenge arises in training deep models, especially for wildlife datasets, due to their limited size, noise, and imbalanced nature. However, a well-optimized algorithm can improve convergence and stability, reduce overfitting, and better navigate the nonconvex loss landscape. It is well known that in wildlife recognition tasks where misclassifications may impact conservation decisions, the optimizer's efficacy is crucial.

Adam, an optimizer for CNNs, shows better convergence and lower validation loss, indicating that it is better suited for specific wildlife classification tasks. This indicates that Adam is well-suited for wildlife classification tasks [7]. In terms of classification accuracy, training efficiency, and loss, Adam outperforms RMSP. This is due to its ability to combine momentum with an adaptive learning rate. A modified Adam optimizer was proposed by Reyad *et al*. [8] to improve performance and address the weaknesses of Adam, including overfitting, overparameterization, and premature convergence. This was done by incorporating a nonlinear decay factor and a laser-widening mechanism. The modified Adam consistently outperformed the standard Adam across accuracy, convergence, speed, and stability.

A study focused on animal behavior classification using wearable Artificial Intelligence on Things (AIoT) was conducted by Arablouei *et al*. [9]. The model was implemented on the embedded systems of AIoT devices and used as a real-time feedback and control mechanism for monitoring livestock. The classification system acts as a control system for data acquisition and communication, transmitting only classified behaviors rather than raw data.

This study offers embedding features: learning within the model itself and deploying it as a control-optimized Artificial Intelligence (AI) system suitable for livestock monitoring, especially in real-world conditions. In a deep learning model, an optimizer can be considered as a control system that determines the best inputs to achieve desired system performance. Another study focused on an effective real-time insect identification system that leverages mobile smart devices [10]. It is based on the YOLOv5-S model, which features a compact design and is best suited for devices with limited hardware resources, offering efficient computing time. The system achieved classification accuracies of 70.5% and 42.9% with mean Average Precision (mAP) scores of 0.5 on the Insect10 and large-scale IP102 datasets, respectively.

Based on the previous information, this study aimed to find the effect of an imbalanced dataset on the RMSProp and Adam algorithms. This paper also proposed data-balancing techniques and enhancements to the RMSProp and Adam algorithms as optimizers to boost the model's performance.

These balancing techniques for imbalanced datasets will address the condition in real wildlife data, where factors such as animal appearance frequency, seasons, species count, movement, and weather and lighting conditions affect data collection. This study extends existing work by evaluating proportional and reproducible augmentation strategies as a balancing technique. By focusing on imbalance techniques, this novel contribution is intended to complement prior research and address the challenges of working with wildlife. The enhancements will cover the learning rate and hyperparameters to improve classification accuracy on a dataset of wildlife images, either full or partial.

This article is organized as follows: Section II provides a literature review; Section III discusses the research methodology, optimization, balancing techniques, and the dataset; Section IV presents the results and analysis of implementing the optimization techniques described in Section III; and Section V summarizes the conclusions of this work.

II. Literature Review

Studies in animal species recognition have used deep CNNs to classify small herpetofauna species such as snakes, lizards, and toads [11]. A comparison of a self-trained CNN and transfer learning (VGG16 and ResNet50) revealed that transfer learning outperformed the CNN, with toads being accurately classified, whereas lizards were more difficult to classify. Due to the imbalanced dataset, augmentation and balancing techniques were integrated. As control mechanisms, early stopping and dropout were applied to reduce overfitting and optimize the performance.

Combining deep CNN architecture with adaptive control-based retraining-maintained robustness under dynamic environmental conditions [12]. This was achieved by implementing a two-stage deep learning framework made for wildlife monitoring. The first stage is the Animal Classification System (ACS), which uses an Xception CNN to distinguish between animal and non-animal images captured by trail cameras. The second stage is the Bird Detection System (BDS), utilizing a faster R-CNN with a ResNet-50 backbone to localize bird species. With Stochastic Gradient Descent with Momentum (SGDM) as the optimizer and a learning rate of 0.001, the ACS reached 99% sensitivity and 96% specificity. While BDS achieved 94% sensitivity and 93% specificity. Due to the imbalance between the animal and no-animal classes,

which could affect model performance, this study aimed to develop a dataset by applying augmentation techniques. Animal images were horizontally flipped and added to the selected images. Variations of the background image were also incorporated into the dataset for the No-animal class. By applying this augmentation technique, the issue of imbalanced data was solved.

A comprehensive analysis and comparison of various deep neural network architectures, such as ResNet50 and DenseNet, applied to wildlife recognition using Infrared (IR) images was conducted [13]. As opposed to commonly found datasets that rely on visible imaging (optical) datasets and may be hindered by darkness and foggy conditions, infrared photos of Slovak wildlife do not have such problems. Infrared imaging captures the thermal signatures of animals for subsequent recognition. Using IR imagery can enhance the robustness of wildlife recognition performance. This comparative study suggests that CNN architectures can address challenges in wild animal recognition, especially in infrared images. Furthermore, this study also suggests that data augmentation strategies should be used to improve the performance of the wildlife recognition model.

A summary of the supporting research is shown in Table I.

The right optimization algorithm can significantly boost a model's performance, particularly in image classification, including animal classification. Nine optimization algorithms were compared, including Vanilla SGD, SGD with momentum, SGD with Nesterov Momentum, AdGrad, RMSProp, Adam, Adadelta, Adamax, and Nadam [14]. Adam achieved the best performance with a deeper/wider CNN, achieving 91.1% accuracy on the cats-and-dogs dataset. Balancing the depth and width of the architecture with the optimizer is essential for efficient convergence and high performance.

TABLE I. SUMMARY OF SUPPORTING RESEARCH

Author(s)	Paper Title	Aims	Deep Learning Model	Optimizer	Performance	Dataset	Novelty
Kartowisastro and Latupapua [7]	A Comparison of Adaptive Moment Estimation (Adam) and RMSProp Optimisation Techniques for Wildlife Animal Classification Using CNNs	To compare Adam vs. RMSProp optimizers in wildlife classification tasks	CNNs (DenseNet-121, ResNet-50, AlexNet)	Adam, RMSProp	Best: Adam + ResNet-50 (80.66%), DenseNet-121 (up to 100%)	Serengeti camera-trap dataset: 46,841 images, 11 classes	Demonstrates optimizers as control mechanisms in CNN training.
Islam *et al*. [11]	Animal Species Recognition with Deep CNNs from Ecological Camera Trap Images	To identify species from ecological camera trap images	Deep CNNs (ResNet variants)	Adam	>90% accuracy	Camera-trap datasets (Texas ecological monitoring, thousands of images)	Focus on ecological monitoring under real-world conditions.
Moallem *et al*. [12]	An Explainable Deep Vision System for Animal Classification and Detection in Trail-Camera Images with Automatic Post-Deployment Retraining	To classify & detect animals with adaptive retraining	Two-stage: Xception CNN (classification), Faster R-CNN (bird detection)	SGD with Momentum (SGDM)	ACS: 94% (day), 98% (night); BDS: 94% SENS, 93% SPEC	TPWD trail-camera dataset (day/night images)	Introduces control-inspired retraining mechanism (RTI & SSIM).
Dogo *et al*. [14]	On the Relative Impact of Optimizers on CNNs with Varying Depth and Width for Image Classification	To evaluate effect of optimizers across different CNN sizes	CNNs (varying depth/width)	-	Accuracy varies across depth; Adam & RMSProp generally stable	CIFAR-10, MNIST, ImageNet	Highlights how optimizers influence CNN scaling.
Luo *et al.* [15]	An Efficient Visual Servo Tracker for Herd Monitoring by UAV	To develop visual tracking system	CNN, YOLOv7 (object detection), Deep SORT (object tracking)	-	Mismach between target bounding box and the setpoint.	Github: YOLOv7-DeepSORT.	Visual servoing in. realtime tracking system with dynamic target.

The optimizer acts as a control system that regulates weight, balancing convergence speed and stability. The findings confirm that Adam consistently outperforms RMSPRop. Combining ResNet50 with Adam at a learning rate of 0.001, achieving 80.66%. RMSProp achieved 77.44%. In this discussion, optimizers can be viewed as control systems that govern weight updates, balancing training speed and stability.

The following is a chronological overview of the previously-mentioned optimization methods, including their first appearances:

(1) SGD: The concept of stochastic approximation was introduced by Robbins and Monro [16] in 1951. However, for neural network training, SGD (mini-batch updates) was recognized in the 1980 s and 1990 s [17]. SGD replaces full-batch gradient with incremental, noisy updates to scale learning over large datasets.

(2) SGD with momentum was formalized by Polyak [18] and later implemented in neural networks by Rumelhart *et al.* [19]. Afterward, the impact of deep networks became well known for improving convergence by smoothing oscillations

and accelerating progress across multiple shallow valleys [20].

(3) Adaptive Gradient (AdaGrad) was first introduced by Duchi [21]. AdaGrad scales the learning rate for each parameter proportional to the square root of the cumulative squared gradient. Although AdaGrad is effective for handling sparse features, it still has a weakness: its step size tends to decay quickly when applied to dense data [22].

(4) Root Mean Square Propagation (RMSProp) was promoted via Geoffrey Hinton's lecture notes [22]. RMSProp upgrades AdaGrad by substituting its unbounded accumulation with an exponentially decaying average of square gradients; in that way, it avoids an excessively small learning rate during training.

(5) Adaptive Moment Estimation (Adam) was first published in late 2014 [23] and presented at ICLR in 2015. The Adam optimizer combines RMSProp's adaptive scaling with momentum (first-moment estimation) and bias correction and has become a widely recognized standard optimizer in deep learning methods [22].

The method described outlines a path from simple incremental updates to smoothing via momentum to per-parameter adaptive learning, and finally to hybrid methods that balance adaptivity and stability. Overall, the chronology of major deep learning optimization methods is summarized in Table II.

These optimization strategies have facilitated rapid progress in deep learning, particularly with RMSProp and Adam, which are widely favored in CNNs. With strong stability and adaptiveness, RMSProp and Adam have given a significant advantage in image classification [22]. The recognition output can be used as input for real-time image-based visual servoing, as Burlacu *et al.* [24] present a real-time visual servoing system equipped with an eye-in-hand camera. It integrates image processing, control computation, and robot motion in a single architecture. This integration can be used to collect images of wildlife and address the challenges posed by imbalanced datasets.

TABLE II. SUMMARY OF SUPPORTING RESEARCH

No.	Major Optimization Methods		
	Optimizer	**First Introduced**	**Key Feature**
1	Stochastic Gradient Descent (SGD)	Robbins and Monro [16]	Random sampling of mini-batches
2	SGD with Momentum	Polyak [18]; applied by Rumelhart *et al.* [19]	Adds inertial memory to gradients
3	AdaGrad	Duchi *et al.* [21]	Adaptive learning rate based on past gradients
4	RMSProp	Ruder [22]	Decay-based averaging of squared gradients
5	Adam	Kingma and Ba [23]	Momentun, RMS Prop and bias correction

III. MATERIALS AND METHODS

A. Research Method

This research analyzed the classification of animal images in three CNN architectures, including AlexNet, ResNet, and DenseNet, and focused on enhancing the optimization algorithm. The research methodology adopted is presented in Fig. 1.

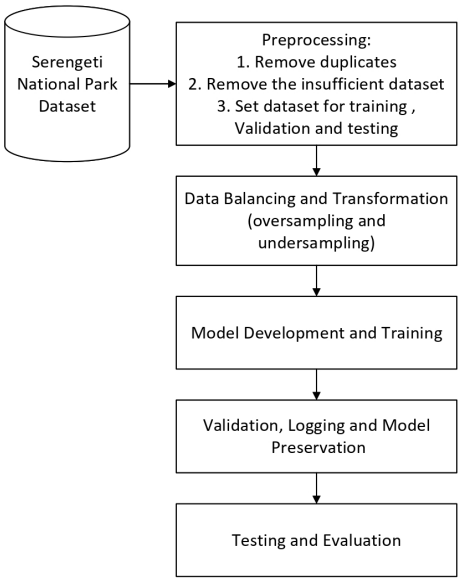


Fig. 1. Research methodology.

Data preprocessing: The method begins with a preprocessing stage consisting of three primary steps. The first step is to remove duplicate images, ensuring that identical samples are excluded to prevent redundancy and potential bias in the learning process. The second step involves verifying the number of images available for each class. Any class that does not meet the minimum threshold is excluded to maintain data quality and consistency across the dataset. The final step in the preprocessing stage is to partition the dataset into three subsets: training, validation, and testing.

From the total dataset, 80% of the images were allocated to the training and validation subsets, with 90% used for training and 10% for validation. The remaining 20% of the dataset was allocated for testing purposes.

Data balancing and transformation: Oversampling and undersampling techniques were applied to address the class imbalance. Oversampling methods, e.g., creating synthetic images, were used for minority classes, and random transformations such as flipping and rotation were also used. The goal was also to capture intraclass variation, so datasets such as buffalo in different poses or lighting conditions were used. On the other hand, under-sampling (reduction) was performed for the majority classes as needed to ensure unbiased class representation. The data transformation was implemented in two distinct pipelines. The training pipeline applied augmentation techniques, including flipping, rotation, and adjusting brightness and contrast, thereby enhancing model generalization. On the contrary, validation and test sets were only resized by

zooming to maintain evaluation consistency. Labels were encoded to map each image to its corresponding class index.

Model development and training: In this stage, hyperparameters were defined, including the CNN architecture, optimizer, learning rate, batch size, number of epochs, momentum, and weight decay. These hyperparameters were set before training as baselines and were critical for regulating the learning process. Next the pretrained CNN model was integrated to enable transfer learning, allowing efficient feature extraction from the dataset. Most layers of the pretrained model were halted to retain prior knowledge, while the final classification layer was replaced to match the number of target classes. To ensure efficient and effective model training, an optimizer was selected by determining how the model's weights were updated to minimize error. During training, a learning rate scheduler was configured to dynamically adjust the learning rate. When a loss stagnates, which indicates the model's performance is no longer improving, the scheduler automatically reduces the learning rate. This was to enhance training stability and improve generalization and performance.

Validation, logging, and model preservation: After each epoch, the model is evaluated on the validation set, and loss and accuracy metrics are computed. The values for each setting were recorded to facilitate analysis of training progress and identify the best-performing model. A graph was created to visualize training and validation accuracy and loss across epochs. These visualizations were used as a reference in reporting and analysis.

Testing and final deployment: After training finished, the final model was evaluated on the test set to assess generalization performance. Performance metrics, such as precision, recall, and F1-Score, were calculated and visualized by using a confusion matrix heatmap. The trained model was indexed for future inference and experimentation, ensuring reusability without retraining.

B. Dataset

A wildlife dataset was collected in the Serengeti National Park in Tanzania, Africa, which is a World Heritage Site. Serengeti National Park is also known as the home of the world's largest unaltered animal migration, spanning 1.5 million hectares of savannah. The data is open access and available in the Snapshot Safari 2024 Expansion dataset [25]. The total number of images we extract from the dataset is 41,762, distributed across 7 classes. However, not all image data consist of complete depictions of animals; some images capture only specific body parts, distant views of animals, grayscale or landscape images, or images with low visual clarity. In addition, the available images are not evenly distributed across the 7 classes, resulting in class imbalance, which constitutes one of the key issues addressed in this study. The details of the image dataset are in Table III.

The dataset taken from Serengeti National Park in Tanzania, Africa, as one of the world's heritage sites, is imbalanced. The nature of the imbalance in this dataset is not just about quantity; it is also about visual similarity. Some rare species might be visually similar to more common ones (e.g., different antelope species). This means that balancing techniques must not only provide more "data points" for the rare class but also help the model learn discriminative features. Rectifying this imbalance generally falls into two categories: a) data-level methods, e.g., altering the dataset itself; b) algorithm-level methods, e.g., improving or modifying the learning process. In this research, a balanced dataset is created using a data-level method.

TABLE III. DATASET OF WILDLIFE ANIMAL IMAGE

No.	Dataset	
	Class	Total
1	Thomson's Gazelle	25,012
2	Grant's Gazelle	4,139
3	Zebra	3,882
4	Guinea Fowl	3,152
5	Warthog	2,546
6	Hartebeest	1,884
7	Bufallo	1,147

As the basis of the research, we conducted pre-processing. The preprocessing data was conducted before the balancing techniques were applied. This pre-processing was conducted by removing images that met the following criteria:

(1) The wildlife subject was not visible in the image;
(2) The wildlife subject was only partially visible in the image;
(3) The wildlife subject did not match the assigned class label;
(4) The image was a duplicate or near-duplicate that did not add meaningful visual information.

The results of pre-processing are as follows:

To address the issue of imbalanced data, we conducted two approaches. The first step was to add the images of Warthog, Hartebeest, and Buffalo, bringing the total to 1,100 images per class. The second one was to randomly pick images of Grant's Gazelle, Zebra, and Guinea Fowl, for a total of 1,100 images per class. The augmented data to be added to the Warthog class consisted of 198 images; the Hartebeest class had 140 images, and the Buffalo class had 423 images (see Table IV, Experiment A).

TABLE IV. DATASET OF SELECTED WILDLIFE ANIMAL IMAGES

No.	Selected Dataset			
	Class	Experiment A	Experiment B	Experiment C
1	Thomson's Gazelle	1,130	1,130	1,130
2	Grant's Gazelle	2,069	2,069	1,100**
3	Zebra	1,992	1,992	1,100**
4	Guinea Fowl	1,821	1,821	1,100**
5	Warthog	902	1,100*	1,100*
6	Hartebeest	960	1,100*	1,100*
7	Bufallo	677	1,100*	1,100*

* data augmentation, ** data reduction.

The data augmentation approach is shown in Table V.

The augmentation approach was divided into four techniques, the first of which was a random flip. The

purpose of a random flip was to simulate different camera angles. The second approach was rotation by ±15 degrees; this technique was used to simulate camera tilt. The third approach involved random zoom-in/out on the required images, with a 15% zoom-in and a 5% zoom-out. The purpose was to see the variation of subject size. The last approach was to adjust brightness or contrast randomly to learn lighting variations.

The dataset was divided into training, validation, and testing subsets, with 80% of the images used for training and 20% for testing; the training portion was then split into 90% training and 10% validation. This strategy ensures that the model is consistently evaluated on unseen data from the same distribution, ensuring generalizability within the dataset.

TABLE V. DATA AUGMENTATION APPROACH

No.	Selected Dataset			
	Technique	Description	Purpose	Composition
1	Random Flip	Horizonal/vertical mirroring	Simulates different camera angles	30%
2	Rotation	±15-degree rotation	Simulates camera tilt	30%
3	Zoom	Random zoom in/out	Variative subject size	Zoom in: 15%, Zoom out: 5%
4	Brightness	Random brightness/contrast adjustment	Vary lighting	20%

C. Concept and Principles of Wildlife Animal Recognition

Dealing with a noisy dataset, such as wildlife in a conservation area like Serengeti National Park, makes it impossible to evaluate performance using an accuracy figure. For the 7-class wildlife problem, rather than examining the 7×7 confusion matrix for each element in detail, an alternative approach is used: a small, carefully selected matrix provides insight into comprehensive, robust model performance.

1) Metrics for wildlife recognition

The overall accuracy value provides a general overview and gives an intuitive sense of performance, since it considers the proportion of total correct recognitions. In principle, it is useful as a sanity check for assessing performance, although it may yield misleading results when data imbalance occurs. Accuracy can be high even when the model is biased. For a confusion matrix, the overall accuracy should be obtained.

Macro F1-Score provides the "true" overall performance since it balances precision and recall. Basically, it is the unweighted average of the F1-Scores for all individual classes. Hence, it gives equal weight to each class, regardless of class size. In principle, this is a primary metric for balanced performance. It is used to assess how well this model performs across all classes (animal species). Overall, the Macro F1-Score directly measures the effectiveness of the implemented balancing techniques.

Buffalo had the smallest dataset. Hence, Buffalo Recall is selected, as it provides information about the actual buffalo images (smallest size) that were correctly recognized. In principle, it is a proxy for the performance of the most vulnerable class.

In any imbalanced dataset, the model's behavior on the smallest class is the most sensitive indicator of bias. If the Buffalo recall is low, the model is considered to have failed, regardless of other metrics. It may also quantify the performance improvement by the balancing technique.

Zebras have a highly unique and consistent visual pattern (black and white stripes). This, in turn, makes them relatively easier for CNNs to learn and recognize as belonging to other animals, such as buffaloes or gazelles, which share similar brown colors. This is the rationale behind looking at Zebra Recall.

Unlike the buffalo class, which has a small dataset, the zebra class has a relatively large dataset (1,992 images) that needs to be reduced (downsampled) to 1,100 images. Zebra Recall reflects the proportion of actual zebra images that were recognized correctly. Since the zebra dataset is downsampled, zebra recall is a proxy for the potential cost of balancing. Putting together these four metrics gives a well-rounded evaluation, where Overall Accuracy provides a reality/sanity check on the raw number of correct guesses, the Macro F-Score gives the overall success, Buffalo Recall provides performance for the most vulnerable class (small dataset), and Zebra Recall gives a measure for potential loss of balancing.

2) Optimization methods

The optimization methods used for this study were RMSProp and Adam. The RMSProp optimization method adapts learning rates by computing the moving average of the root mean square of the gradients, enabling effective performance in both online and non-stationary learning scenarios. The formula of this moving average, E, is described in Eq. (1) as follows:

$$E\left[\Delta\theta^2\right]_t = \gamma E\left[\Delta\theta^2\right]_{t-1} + \left(1-\gamma\right)\Delta\theta_t^2 \tag{1}$$

where γ is a constant set to 0.9, and ε is a small number.

The Root Mean Squared (RMS) error value of the corresponding parameter update is then described by following:

$$RMS\left[\Delta\theta\right]_t = \sqrt{E\left[\Delta\theta^2\right]_t + \varepsilon} \tag{2}$$

$$E\left[g^2\right]_t = 0.9E\left[g^2\right]_{t-1} + 0.1g_t^2 \tag{3}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E\left[g^2\right]_t + \varepsilon}} g_t \tag{4}$$

With this, RMSProp can adaptively adjust learning rates while avoiding excessively small values.

Adam is a stochastic optimization algorithm that relies only on first-order gradients and requires little memory. It maintains gradient (m_t) and squared gradient (v_t) averages across iterations. The respective update rules, with decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$, are defined as Eqs. (5)–(7):

$$g_t = \nabla_\theta J(\theta_t) \tag{5}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) g_t \tag{6}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \tag{7}$$

where g_t represents the gradient at step t, m_t and v_t represent the first moment and the second moment (variance), respectively.

The results of the above formulas are refined using bias correction, yielding the corrected first and second moments, and the final parameter update rule is then computed as following:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{8}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{9}$$

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{10}$$

3) Deep learning models

This study used a CNN architecture based on both simple and relatively complex architectures. This accounts for the need for fast computation while remaining robust enough to handle challenges in wildlife conservation. With this perspective, the implementation of the visual servoing system will be supported. Hence, this work investigates any approach, whether simple or complex, to architecture. AlexNet, DenseNet, and ResNet were selected.

AlexNet, as a simpler, older architecture, has lower capacity. It might saturate quickly across most classes and struggle to learn complex features of rare species. Balancing techniques will be critical to force its limited parameters to focus on the minority classes. On the other hand, DenseNet and ResNet, which are high-capacity models, are powerful enough to memorize the entire dataset, including rare classes, when trained correctly. Without balancing, they might still take the "easy way out" and be biased towards the majority classes. With balancing, their powerful feature extraction can be fully leveraged for the rare species. DenseNet's feature reuse might be particularly effective in learning from limited examples.

During the whole experiment with 60 epochs, Learning Rate (LR) values varied from $1\times e^{-3}$ to $1\times e^{-5}$, since a slow learning rate may lead to training divergence or oscillation, but on the other hand, for oversampling, it has potential for a less spiky gradient and may lead to faster convergence. Higher learning rates have slower training, and the model may not train sufficiently in 60 epochs.

4) Pathway to the Image-Based Visual Servoing (IBVS) system

Once the wildlife animal recognition capability is obtained, this provides a sight (feedback by a visual sensor) for an IBVS system. A complete diagram of the IBVS is shown in Fig. 2. This IBVS creates an automatic control system, namely an autonomous wildlife "cameraperson".

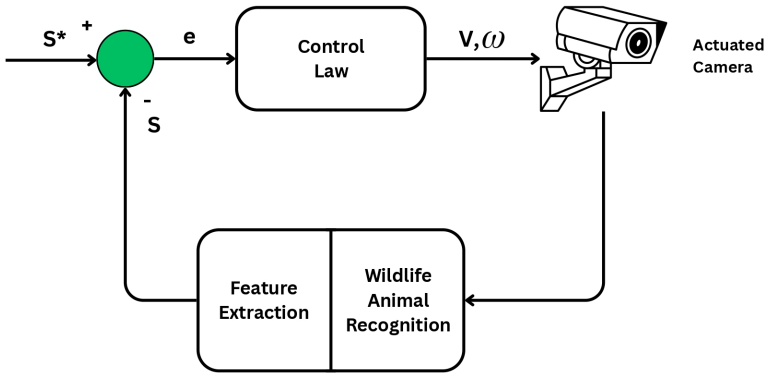


Fig. 2. Image-Based Visual Servoing (IBVS).

IBVS is part of the control strategy the system uses. IBVS means that the error signal that drives the actuator to which a camera is attached is calculated directly in the 2D image space. The overarching system and goal are to actively monitor and track wildlife without human intervention. IBVS is executed as follows:

(1) Wildlife animal recognition: The core task of identifying and classifying animals in an image. The animal's rough location is known through bounding boxes. The CNN architecture is used to perform wildlife recognition.
(2) Through feature extraction, the system provides information about the precise point in the image (Output: Centroid Coordinates).
(3) Control law finally ensures that the camera tracks the precise location of the wildlife.

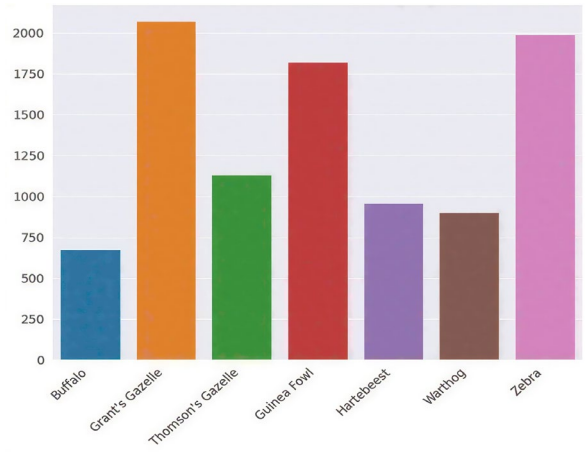
In short, CNN performs wildlife recognition, and the output of this process is used as input to an IBVS controller. The integration of these technologies creates a visual servoing system for wildlife conservation (an autonomous wildlife camera operator). This manuscript, however, focuses more on the wildlife recognition component.

IV. Results and Discussion

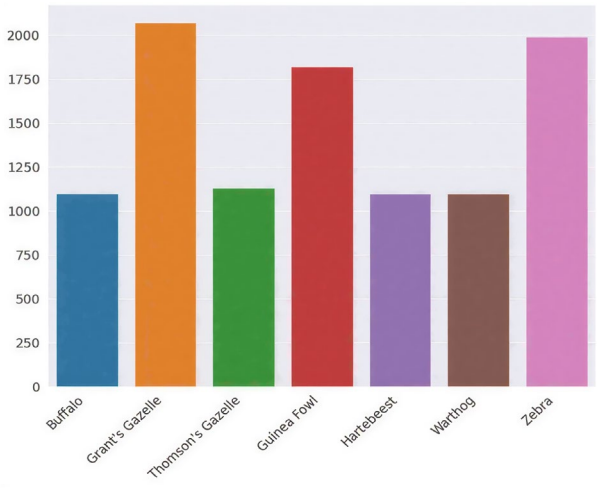
A. Experimental Setup and Result

Three experimental setups were conducted: Experiment A with seven original dataset classes, Experiment B with four original dataset classes and three augmented dataset classes, and Experiment C with one original dataset class, three augmented dataset classes, and three reduced dataset classes (Table IV and Fig. 3).

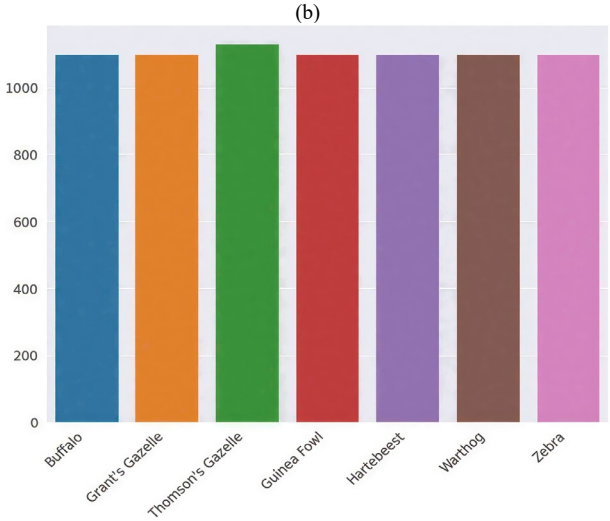
Certain parameters were fixed across all experiments, including a batch size of 32, 60 training epochs, and cross-entropy as the loss function for multiclass classification. Accuracy was then evaluated using three different models, namely DenseNet-121, ResNet-50, and AlexNet, with Adam and RMSProp as optimizers and at learning rate values of 1.0×10^{-3}, 1.0×10^{-4}, and 1.0×10^{-5}. Table VI presents the best learning rates identified for each experiment, along with the corresponding overall accuracy results.



(a)



(b)



(c)

Fig. 3. Experiment A, B, and C dataset distribution. (a) Experiment A dataset. (b) Experiment B dataset. (c) Experiment C dataset.

TABLE VI. Summary of Overall Accuracy, Macro F1-Score, Buffalo Recall, and Zebra Recall for Adam and RMSProp from Experiments A, B, and C

Architecture	Parameters						
	Optimizer	Experiment	Best LR	Overall Accuracy	Macro F1-Score	Buffalo Recall	Zebra Recall
AlexNet	Adam	A	1.0×10^{-3}	0.7262	0.7613	0.9028	0.8417
AlexNet	RMSProp		1.0×10^{-4}	0.7251	0.8118	0.9097	0.7789
DenseNet	Adam		1.0×10^{-3}	0.7068	0.6853	0.8403	0.6935
DenseNet	RMSProp		1.0×10^{-4}	0.7047	0.6904	0.8264	0.7211
ResNet	Adam		1.0×10^{-3}	0.6665	0.7416	0.7014	0.7588
ResNet	RMSProp		1.0×10^{-3}	0.6770	0.7339	0.6736	0.7789
AlexNet	Adam	B	1.0×10^{-4}	0.6571	0.7708	0.7360	0.7384
AlexNet	RMSProp		1.0×10^{-4}	0.6489	0.7434	0.6954	0.6921
DenseNet	Adam		1.0×10^{-3}	0.6935	0.7040	0.7513	0.6944
DenseNet	RMSProp		1.0×10^{-3}	0.6877	0.6928	0.7513	0.6644
ResNet	Adam		1.0×10^{-4}	0.7139	0.7612	0.6447	0.8403
ResNet	RMSProp		1.0×10^{-4}	0.6891	0.7467	0.6294	0.7847
AlexNet	Adam	C	1.0×10^{-4}	0.6481	0.7725	0.8565	0.5892
AlexNet	RMSProp		1.0×10^{-3}	0.6766	0.7749	0.9283	0.6763
DenseNet	Adam		1.0×10^{-4}	0.6947	0.7392	0.8072	0.6390
DenseNet	RMSProp		1.0×10^{-4}	0.6876	0.7398	0.7982	0.6473
ResNet	Adam		1.0×10^{-3}	0.6617	0.7737	0.8206	0.7635
ResNet	RMSProp		1.0×10^{-3}	0.6617	0.7737	0.8206	0.7635

As previously discussed, the 7 class animal species and balancing techniques were implemented to gain insight into the recognition performance, where zebra and buffalo were of concern due to their uniqueness and minority class, respectively.

In early-stage experiments with various epoch settings on AlexNet, ResNet, and DenseNet, each architecture exhibited distinct characteristics. AlexNet showed relatively good results with fewer epochs than ResNet and DenseNet. This situation was impacted by the number of layers of each architecture. AlexNet has significantly fewer layers than ResNet and DenseNet. Figs. 4–6 present preliminary experiments on three architectures using the Adam optimizer with a learning rate of 1.0×10^{-5}. The overfitting stopper was configured to detect the best epoch for each architecture. Figs 4–6 show that AlexNet, ResNet, and DenseNet achieve the best results at 28, 89, and 100 epochs, respectively. These preliminary experiments demonstrated that the number of epochs must be treated as one convergence point among the three. Thus, the 60 epochs were chosen to fulfill the convergence.

Four metrics are paramount for gaining insight into the system: overall accuracy, F1-Measure (a harmonized measure of Precision and Recall), Zebra Recall, and Buffalo Recall. These are presented in Table VI.

In order to provide a complete perspective, 3 types of analysis were conducted:

(1) Analysis by experiment (balancing impact).
(2) Analysis by architecture.
(3) Analysis by architecture and optimizer pair.

Lastly, the connection between wildlife recognition and IBVS performance is presented.

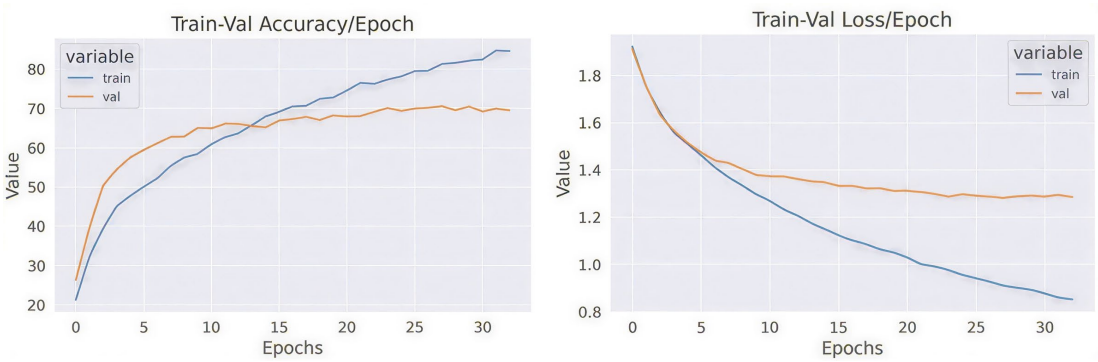


Fig. 4. Training and validation graph of AlexNet with best epochs was 28.

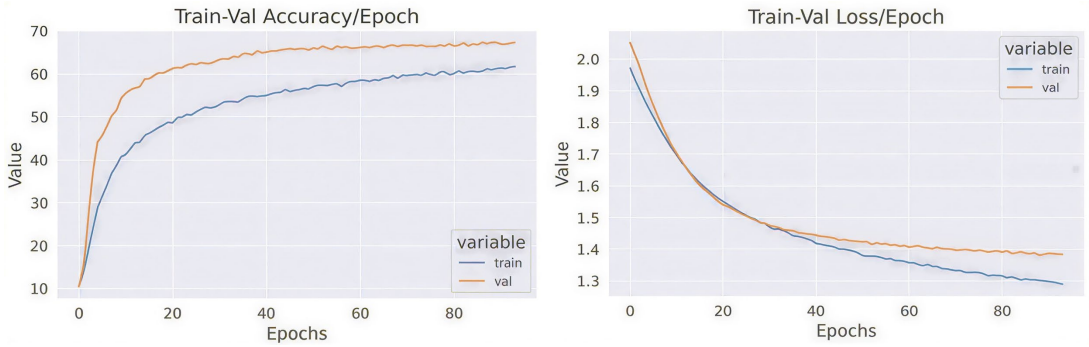


Fig. 5. Training and validation graph of ResNet with best epochs was 89.

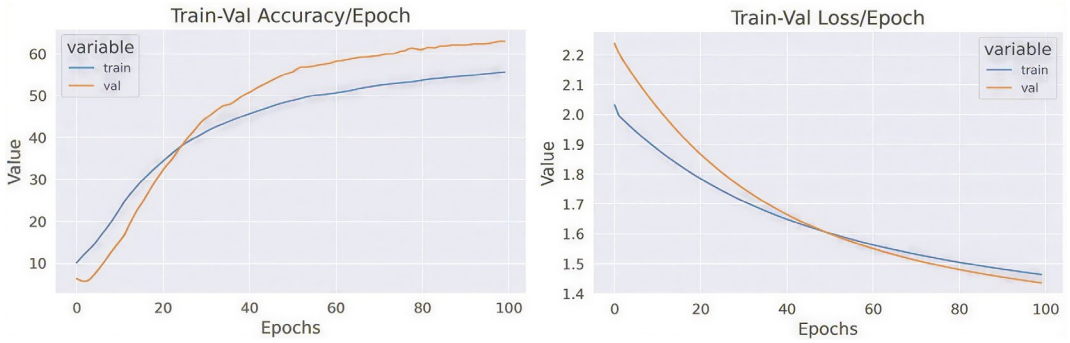


Fig. 6. Training and validation graph of DenseNet with best epochs was 100.

B. Analysis by Experiment

1) Experiment A (original imbalanced dataset)

In experiment A, the best-performing model was AlexNet with the RMSprop optimizer and a learning rate of $1e^{-4}$. This experiment produced a key metric, Macro F1-Score = 0.8118.

Analysis and Justification with respect to Balancing Impact—In the challenging, noisy landscape of the imbalanced dataset, the combination of a lower-capacity

model (AlexNet) and a more stable, conservative optimizer (RMSprop) proved superior. AlexNet, with its limited parameters, was less prone to overfitting the spurious correlations of the majority classes. Meanwhile, RMSprop's gradient normalization prevented it from being overly swayed by the dominant classes, allowing it to find a more generalizable solution that still performed remarkably well on the minority class, buffalo (Recall: 90.97%). The high-capacity models (DenseNet, ResNet) were more easily biased by the imbalance, as they had the capacity to simply memorize the majority classes, which was the path of least resistance for reducing loss. The accuracy and loss for Experiment A are depicted in Fig. 7.

2) Experiment B (dataset with augmentation)

In experiment B, the best-performing model was AlexNet with the Adam optimizer and a learning rate of $1e^{-4}$. This experiment produced a key metric, Macro F1-Score = 0.7708.

Analysis and justification with respect to balancing impact—Experiment B created a unique hybrid dataset: the minority classes (buffalo, hartebeest, warthog) were boosted via augmentation, but the original, large majority classes (Grant's Gazelle, Zebra, Guinea Fowl) were left intact.

(1) The most effective sweet spot for AlexNet: this partially balanced state was the perfect environment for it. It was no longer as severely imbalanced as Exp A, so AlexNet's lower capacity was not a major handicap. However, the problem was not as uniformly balanced as in Exp C, meaning AlexNet's simplicity still gave it an advantage in converging efficiently to a good solution within the 60-epoch budget. It effectively used the new augmented data for the minority classes without being overwhelmed by the still-complex task of learning from the full, unreduced majority classes.

(2) ResNet's challenge: Despite its higher capacity, ResNet was still in a transitional phase in Exp B. The remaining imbalance and data volume from the majority classes meant it had not yet reached the ideal conditions (provided by Exp C) to fully leverage its power. It was likely still learning and had not yet converged on a solution that could surpass the efficiency of the trained AlexNet.

(3) Optimizer synergy: The Adam optimizer performed well with AlexNet, achieving efficient convergence. The combination was a very strong balance between learning from the newly augmented minorities and the original majorities.

The accuracy and loss for Experiment B are depicted in Fig. 8.

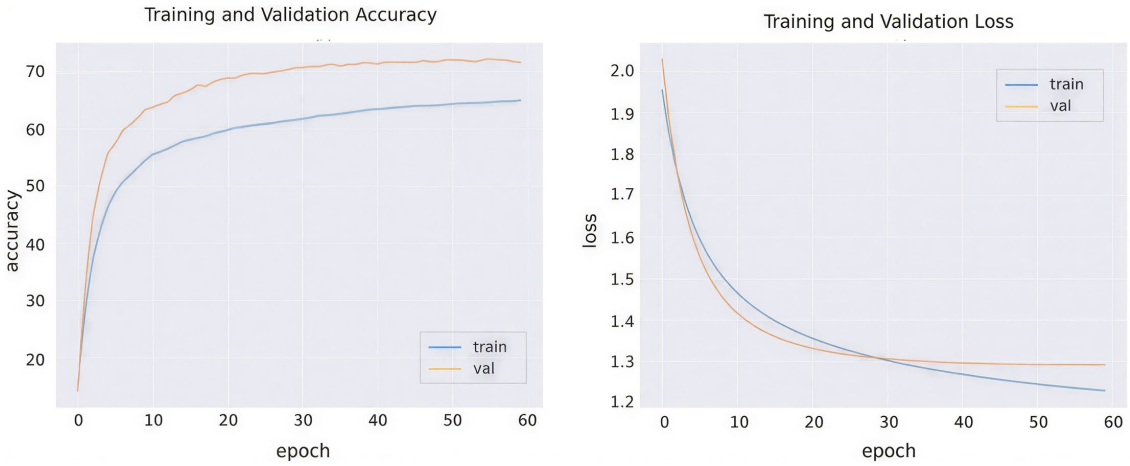


Fig. 7. Training and validation graph experiment A for AlexNet-RMSprop.

Fig. 8. Training and validation graph experiment B for AlexNet-Adam.

3) Experiment C (dataset with a combined augmentation and reduction)

For Experiment C, the AlexNet-RMSprop model achieved the highest Macro F1-Score (0.7749). The ResNet-Adam model achieved a nearly identical score (0.7737). Given the insignificant numerical difference, their performances were considered equal. Therefore, model selection must also consider other requirements, such as inference speed and computational resources.

Analysis and Justification with Respect to Balancing Impact—The perfectly balanced dataset in Experiment C created an ideal, uniform learning environment. This allowed two different optimal solutions to emerge:

(1) AlexNet with RMSprop: The full balancing simplified the learning problem to the point where AlexNet's limited capacity was no longer a hindrance but a benefit. It learned the now-manageable task perfectly, achieving the highest Macro F1-Score of the entire study, driven by an exceptional Buffalo recall (0.9283).

(2) ResNet with Adam: The high-capacity ResNet, guided by the efficient Adam optimizer, also reached its peak performance in this ideal setting. It leveraged its sophisticated feature extraction to achieve very high, balanced performance on both Buffalo recall (0.8206) and Zebra recall (0.7635).

The 0.0012 difference in their Macro F1-Scores is negligible. The choice between them would depend on the secondary requirement: AlexNet for potentially faster inference, or ResNet for its ability to maintain stronger performance on the majority class. The accuracy and loss for Experiment C are depicted in Figs. 9 and 10.

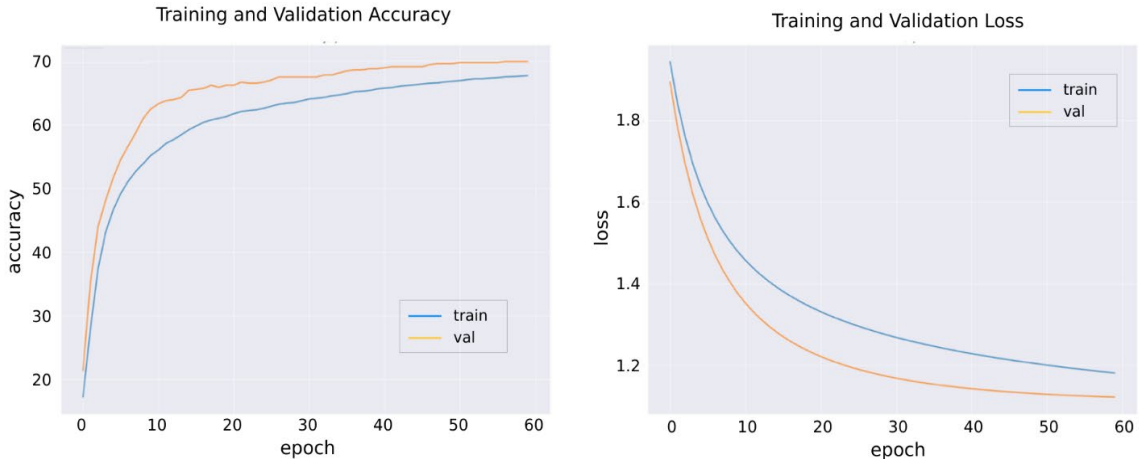


Fig. 9. Training and validation graph experiment C for AlexNet-RMS Prop.

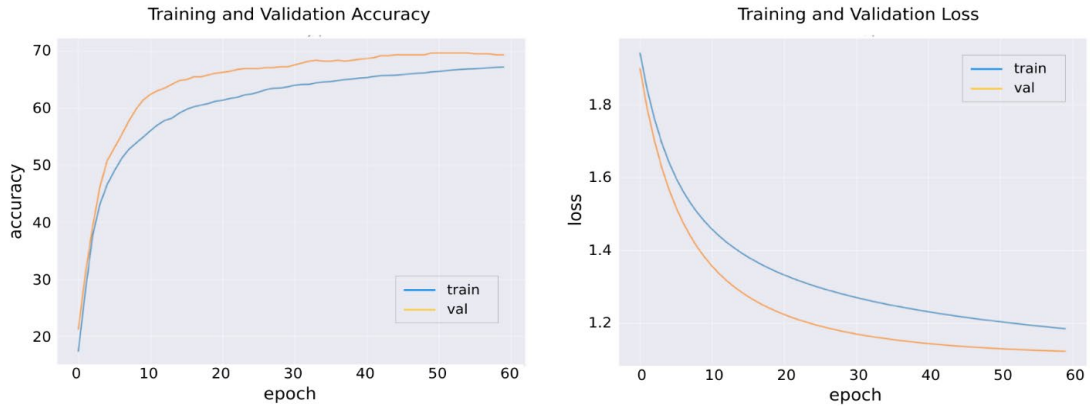


Fig. 10. Training and validation graph experiment C for ResNet-Adam.

4) Overall conclusion of the experiments

(1) Experiment A (Imbalanced): AlexNet + RMSprop—It outperformed the others through robustness and a specialization effect on the minority class.

(2) Experiment B (Data Augmentation): AlexNet + Adam—This configuration wins by efficiently leveraging partially balanced data, hitting the sweet spot in architectural complexity.

(3) Experiment C (Combined Augmentation and Reduction-Full Balance): AlexNet + RMSprop—Combining AlexNet with RMSprop for a perfectly balanced task is ideally suited to its capacity, making it a reliable generalist.

This analysis shows a consistent theme: AlexNet, when paired with the right optimizer, was remarkably effective and efficient for this specific dataset across all balancing regimes. The more complex ResNet architecture only

became a truly competitive alternative under the ideal conditions of full balance in Experiment C.

C. Analysis by Architecture

Analysis by architecture was conducted using the results summarized in Table VI for Overall Accuracy and Macro F1-Score across all experiments. Model capacity, learning rate, and computational burden over 60 epochs were considered when exploring how each architecture performs in wildlife animal recognition.

1) AlexNet

AlexNet is a lower-capacity model with 8 layers and approximately 60 million parameters. The term lower capacity means it has less "learning power" but faster convergence and a stronger built-in regularization effect.

The architecture-based analysis of AlexNet was based on the results from all experiments. In Experiment A, AlexNet demonstrated that its simplicity was an advantage. It avoided overfitting the complex, imbalanced data and, with RMSprop and a Macro F1 of 0.8118, found a stable solution that was coincidentally excellent for the minority class. In Experiment B, the additional data for minority classes provided more to learn from without making the overall problem too complex. It converged efficiently to the best performance observed in this experiment, achieving a Macro F1 of 0.7708 with the Adam method. In Experiment C, the balanced dataset condition allowed AlexNet with RMSProp to achieve its highest and most robust performance, with a Macro F1 of 0.7749.

Architecture performance was then further analyzed with respect to epoch and learning rate. Across all experiments, 60 epochs were sufficient for AlexNet. As a lower-capacity model, AlexNet learned quickly and likely converged well before the 60-epoch limit.

Based on the experimental results, AlexNet performed best with a moderate learning rate of $1e^{-4}$ in most cases, but it also performed effectively with a higher value of $1e^{-3}$ in Experiment C, where the smooth loss landscape allowed for larger, faster steps.

2) ResNet

ResNet is a high-capacity model with approximately 25 million parameters and over 50 layers. This high capacity enables ResNet to learn complex features, but it requires more data and time to converge and avoid simply memorizing the training set. As with AlexNet, the ResNet analysis was conducted throughout all experiments.

In Experiment A, with its high capacity, ResNet suffered from overfitting to the majority classes and failed to learn the minority classes effectively. It was misled by the data imbalance. In Experiment B, augmentation provided a better signal, but the remaining imbalance and data volume from the majority classes meant it was still not in an ideal learning environment. It was improving, and with the ADAM method, it achieved a Macro F1 of 0.7612 but had not yet surpassed AlexNet's performance. Unlike in Experiment A and Experiment B, Experiment C used full balancing, which created ideal conditions for ResNet. The uniform data distribution guided its powerful feature extraction, enabling it to learn a highly balanced and robust model, achieving a Macro F1 of 0.7737, a negligible difference from the best AlexNet Macro F1 of 0.7749.

The analysis from the perspectives of epoch and learning rate revealed that ResNet's higher capacity resulted in a longer convergence trajectory. Conditions with 60 epochs were sub-optimal. Hence, it was still improving when training stopped. More epochs would have allowed it to surpass AlexNet. ResNet typically requires a high learning rate of $1e^{-3}$ to make significant progress within the epoch budget. This suggests it needs longer update steps to navigate its vast parameter space efficiently within the limited time available.

3) DenseNet

Like ResNet, DenseNet is also a high-capacity model but designed for feature reuse, making it parameter-efficient. In experiments A, B, and C, Table VI shows that DenseNet consistently was the lowest performer. Its feature reuse mechanism, while efficient, may have made it more sensitive to the initial imbalanced data distribution in Exp A. It may have struggled to unlearn the early biased features it developed when the dataset was balanced in later experiments. It never truly recovered or adapted as well as the other architecture. Overall, these findings indicate that, although it has theoretical strengths in feature reuse and parameter efficiency, DenseNet may be less suitable than other architectures. Its consistent underperformance suggests that 60 epochs and the chosen learning rate range were not the primary issues. It suggests that the architecture itself was a poor fit for this specific dataset and task.

4) Overall conclusion on architecture

Overall, the experimental outcome suggests that the choice of architecture must be aligned not only with the complexity and nature of the classification task, but also with the practical limitations of the training environment, in particular, the epoch budget and computational burden. For a fixed, limited epoch budget (60 epochs), AlexNet is the superior choice. Its lower capacity is not a limitation but an advantage, allowing it to converge quickly and reliably to a very high-quality solution, especially on balanced data.

DenseNet consistently underperformed in all experiments (the lowest performer). Even after running different learning-rate (hyperparameter) experiments, it did not achieve the best performance in any of the nine configurations (3 learning rates across experiments A, B, and C). Moreover, DenseNet lacks responses to the dataset balancing technique.

The experiments revealed that ResNet and AlexNet showed clear positive responses to the dataset-balancing techniques, whereas DenseNet's improvement was marginal. This condition suggests that its architecture had difficulty leveraging the improved dataset balance. If computational resources and time are unlimited, ResNet is the better long-term investment. Given a fully balanced dataset (Experiment C) and more training time (more epochs), it is expected to eventually outperform AlexNet

and achieve the highest possible accuracy due to its superior feature-learning capabilities.

The key takeaway is that model capacity must be matched to both the complexity of the task and the resources available for training. The experiments show that for the Snapshot Serengeti 7-class problem, AlexNet was often the most efficient and effective tool. ResNet, a high-capacity model, was underutilized and required more epochs, whereas DenseNet was a poor fit for this specific dataset. Hence, ResNet is a promising architecture if time and resources allow for longer optimization trajectories.

Beyond predictive performance and computational load, AlexNet has superior performance given its relatively low Floating-Point Operations (FLOPs), which measure the number of multiplications and additions required for inference (one forward pass), and hence it has low latency. AlexNet typically has 0.7–1.5 billion FLOPs, while ResNet and DenseNet have 3.8–4.1 billion FLOPs and 2.8–3.0 billion FLOPs, respectively. Due to its significantly lower FLOP count, AlexNet can process an image faster than ResNet and DenseNet when implemented on the same hardware (CPU/GPU).

Overall, these findings suggest that model capacity must be carefully matched to both the problem characteristics and resource limitations of training and deployment. For the Snapshot Serengeti seven-class wildlife classification task, AlexNet showed the most efficient and practically effective solution, while ResNet was relatively underutilized, and DenseNet appeared to be a poor fit for the dataset. In the context of real-time Image-Based Visual Servoing (IBVS), in which low-latency inference is crucial, AlexNet offers the strongest practical advantages. The combination of competitive classification performance and a lower computational burden makes it the most suitable architecture when responsiveness and efficiency are as important as accuracy.

D. Analysis by Architecture and Optimizer Pair Right-Sizing Principle

Based on the results summarized in Table VI, the optimal architecture is not the most powerful one in absolute terms, but the one whose capacity best matches the complexity of the learning task.

This is evidenced by AlexNet (a lower-capacity model) achieving top performance because the 7-class, fully balanced recognition task was perfectly "right-sized" for it. ResNet (high-capacity) was underutilized and required more epochs, while DenseNet was a poor fit for this specific dataset.

Optimizer Choice is context-dependent on architecture and data. The results throughout all the experiments showed that there was no single best optimizer. The optimal choice depends on the architecture's capacity and the dataset's balance. Evidence that supports this insight is as follows:

(1) RMSprop performed well with AlexNet, especially on both imbalanced and fully balanced data (Experiments A and C). Its conservative, stable updates were a perfect match for AlexNet's simple landscape.
(2) Adam showed its strength with ResNet as the data became more balanced (Experiments B and C). Its adaptive, momentum-driven updates helped the high-capacity model make more efficient progress through its complex loss landscape within the limited epoch budget.

An investigation of the experiment results revealed that an architecture's superiority was not consistent across all conditions; it depended on the dataset's composition. This is supported by the fact that, for imbalanced data (Experiment A), AlexNet outperforms the others. Its lower capacity prevents severe overfitting to the majority class. For partially balanced data (Experiment B), AlexNet was also the best among these three architectures. It appears that the problem is still not complex enough to fully leverage ResNet's power within a 60-epoch budget. Finally, for fully balanced data (Experiment C), both AlexNet and ResNet are numerically tied in their Macro F1-Scores. Here, the playing field is level. AlexNet efficiently mastered the task, while ResNet, if given more time, has the potential to surpass it.

High-capacity models require a greater burden upfront, not just in data balancing but also in training time (epochs) to realize their potential. Evidence, as presented in Table VI, shows that ResNet's near-parity with AlexNet in Experiment C, despite likely not having fully converged in 60 epochs, indicates a higher performance ceiling. However, achieving that ceiling requires a commitment to longer training, which may not be efficient for all applications.

A decrease in Overall Accuracy, when coupled with an increase in Macro F1-Score, is a positive sign that a model is becoming less biased and more useful for real-world conservation.

The best models in Experiments B and C had lower accuracy than those in Experiment A, but their balanced performance (Macro F1-Score) was far superior. This confirms that accuracy is a misleading metric for imbalanced datasets.

E. Towards Implementation of IBVS

In developing an IBVS system for wildlife conservation, where real-time computation is paramount and a firm operational requirement, the most appropriate configuration is AlexNet with the RMSprop optimizer.

This is based upon some considerations, as follows:

(1) Faster Inference Speed: AlexNet is a significantly smaller and less complex architecture than ResNet. It has about 60 million parameters and 8 layers, whereas a standard ResNet-50 has approximately 25 million parameters and 50 layers, with much more complex residual block operations. This directly enables AlexNet to process frames much faster. In a visual servoing loop, the system must capture an image, process it with a CNN, compute the error, and send a command to the actuators. A shorter processing time per frame allows for a higher control frequency, which is essential for smooth, stable, and responsive tracking of a moving animal. AlexNet provides the low latency required for this task.
(2) Lower Computational Resource Demand: A visual servoing system, especially one deployed on a mobile platform, has limited battery power and computing

hardware. AlexNet's lower computational footprint means it consumes less power and can run efficiently on this constrained hardware. Attempting to run ResNet in real time would require more powerful, power-hungry hardware, which is often impractical for field deployment.

In the context of classical control theory, AlexNet serves as a feedback element. This AlexNet's output is a selected and weighted feature, which is the animal's image coordinate $y(t) = [x, y]^T$. This is an important feedback signal and acts as the current image features. The difference between the desired and current image features yields visual feature errors that IBVS must minimize. The control system performance metrics, such as rise time, steady state error, and stability defined by the damping ratio (ζ), are important to gain insight into how the control system behaves.

The experiments revealed that the lower-capacity AlexNet model with RMSProp and a learning rate of $1e^{-4}$ in Experiment A is optimal for IBVS implementation. The Macro F1-Score of 0.8118 indicates balanced and high recall across 7 wildlife species, and it provides the $y(t)$ feedback signal (current image features) to the IBVS control law. Because AlexNet's architecture provides low inference latency (Δt), it minimizes the impact on phase lag in the control loop, which can degrade relative stability. RMSProp's good convergence yields a model with low output noise (jitter), which prevents the controller from reacting to fake data.

When an animal is recognized, and the system tries to track the animal, a transient state is present in the system, and the transient response is characterized by its natural frequency (ω_n) and damping ratio (ζ). The rise time in this transient state also depends on the damping ratio (ζ) value. In most cases, the critical-damping ($\zeta = 1$) condition is preferable, as it achieves the fastest possible rise to the setpoint with no overshoot.

Steady-state error, the continuous error $e(t)$ after all transients have decayed, is affected by AlexNet in two ways. First, the model's high Macro F1-Score and generalization capability minimize systematic prediction bias. Hence, it also minimizes the use of integral control action, which can affect the damping effectiveness. Second, the CNN's reliability can improve the continuous validity of $y(t)$. In the case of intermittent loss of the $y(t)$ signal, this may cause the integral term of a PID controller to accumulate erroneous corrections, leading to oscillation. AlexNet will make the integral controller function as designed to remove steady-state error.

On the other hand, AlexNet latency affects the damping ratio (ζ) value. Since AlexNet is a low-capacity model and has fewer layers than DenseNet and ResNet, its low latency allows for the use of higher controller gains without destabilizing phase lag that would drive ζ toward zero (increasing overshoot and oscillation). Consequently, this will ease the process of tuning the gain to place the closed-loop poles at $\zeta \approx 1$, enabling the camera to smoothly arrive at the center of an animal. This is a performance advantage attributable to AlexNet. Because the damping ratio, ζ, determines system stability and is modulated by AlexNet performance, introducing AlexNet can improve overall IBVS performance.

The results of this work suggest an improvement over traditional IBVS performance by providing wildlife recognition modules that inform the control law to respond adaptively to animal behavior and are more robust to environmental clutter. A framework showing how traditional IBVS performance can be enhanced by considering AlexNet wildlife animal recognition is depicted in Fig. 11.

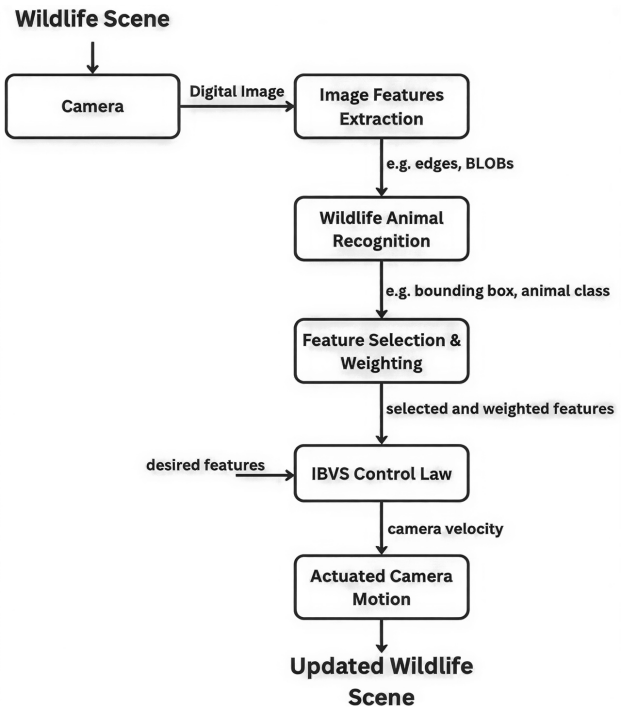


Fig. 11. Enhanced image-based visual servoing framework.

The IBVS Framework is detailed as follows:

(1) Image acquisition and feature extraction

Wildlife scenes are grabbed and converted to digital images by a camera. This digital image is then processed to extract low-level features, such as edges and BLOB analysis results.

(2) Wildlife animal recognition

The trained AlexNet detects and classifies wildlife animals from digital images. This process produces bounding boxes and classified animal classes.

(3) Feature selection and weight

Features and recognition results are combined to select and weight features from the recognized wildlife and deliver them to IBVS. This is to help the controller focus on the animal by giving greater weight to features within the bounding box. This, in turn, reduces disturbances from the background clutter.

(4) IBVS

IBVS maps visual feature errors (the difference between desired image features and current image features) to the camera velocity command. This is governed by the following control law, as shown in Eq. (11).

$$v = -\lambda L + W\left(s - s^*\right) \tag{11}$$

where:
v = velocity which drives the camera movement.
λ = gain for control velocity.
L^+ = pseudo inverse Jacobian.
W = diagonal weighting matrix.
s = current visual features.
s^* = desired visual feature.

The integration of AlexNet with IBVS creates a visual servoing system for wildlife conservation, and the efficacy of an IBVS system for wildlife monitoring is fundamentally constrained by the quality of its visual perception.

High-capacity models like ResNet require more epochs to surpass this AlexNet performance. For robust field deployment, applying the dataset balancing techniques from Experiments B and C to a high-capacity model with more epochs may improve its generalization to field variations.

V. Conclusion

This work provides a definitive, data-driven framework for building effective CNN-based wildlife animal recognizers, explicitly addressing the critical challenge of dataset imbalance. The interplay among dataset balancing techniques, model architecture, and the optimizer is a key factor in success.

While AlexNet with RMSprop achieved the highest Macro F1-Score (0.8118) under 60 epochs of training, it is important to consider the role of model capacity and training time. The high-capacity ResNet model achieved a close score (0.7737) in the same timeframe. This suggests that while AlexNet was well-suited and converged efficiently, ResNet's performance was likely to still increase. Therefore, given a larger training budget (a higher number of epochs), the more powerful ResNet architecture would be expected to outperform AlexNet, reaching a higher final performance ceiling at the cost of increased computational time and resources.

This work underscores that the choice of evaluation metric is critical. Macro F1-Score, coupled with targeted per-class recall, is the correct approach for wildlife animal recognition, revealing truths that overall accuracy cannot capture. This work suggests that AlexNet with RMS Prop is a good choice for real-time visual servoing.

The observation that AlexNet-RMSProp achieves an optimal balance between speed and accuracy provides a foundation for significant future research aimed at developing a robust, real-time IBVS system. Works can be directed to validate the model's performance across a variety of camera-trap datasets, such as those from Caltech, to confirm that it is not overfitting to the Serengeti data.

Subsequently, optimization can be further refined, and modern optimizers, such as RAdam and AdaBound, can be used. An investigation can be conducted to determine whether these methods improve convergence and performance, especially for high-capacity architectures such as ResNet, with balancing techniques implemented in Experiments B and C.

Finally, the most important evaluation of the proposed framework is in its real-time operational performance, on suitable AI hardware platforms. Works can be directed to validate a complete perception-to-action loop using key control metrics such as rise time, settling error, steady-state error, and stability.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Iman Herwidiana Kartowisastro: planned the research design, carried out the literature search and experiments, conducted research analysis, and created and revised the manuscript. Cuk Tho: conducted the literature search and experiments, collected and curated the dataset, assisted with research analysis, and revised the manuscript. Reina Setiawan: conducted program code development, assisted in conducting research analysis, and revised the manuscript. All authors had approved the final version.

Acknowledgment

This research was conducted under the Bina Nusantara University professorship research program. The authors would like to thank Bina Nusantara University for this. Without such an opportunity, this work would not have been possible.

References

[1] D. Tuia, B. Kellenberger, S. Beery *et al*., "Perspectives in machine learning for wildlife conservation," *Nature Communications*, vol. 13, no. 1, 792, 2022. doi: 10.1038/s41467-022-27980-y

[2] O. Debauche, M. Elmoulat, S. Mahmoudi, J. Bindelle, and F. Lebeau, "Farm animals' behaviors and welfare analysis with AI algorithms: A review," *International Information and Engineering Technology Association*, vol. 35, no. 3, 2021. doi: 10.18280/ria.350308

[3] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artif. Intell. Rev.*, vol. 57, no. 4, 99, 2024. doi: 10.1007/s10462-024-10721-6

[4] F. Schindler and V. Steinhage, "Identification of animals and recognition of their actions in wildlife videos using deep learning techniques," *Ecol. Inform.*, vol. 61, 101215, 2021. doi: 10.1016/j.ecoinf.2021.101215

[5] A. Gullapelly and B. G. Banik, "Classification of rigid and non-rigid objects using CNN," *Revue d'Intelligence Artificielle*, vol. 35, no. 4, pp. 341–347, 2021. doi: 10.18280/ria.350409

[6] D. O. Melinte and L. Vladareanu, "Facial expressions recognition for human-robot interaction using deep convolutional neural networks with rectified Adam optimizer," *Sensors*, vol. 20, no. 8, 2393, 2020. doi: 10.3390/s20082393

[7] I. H. Kartowisastro and J. Latupapua, "A comparison of Adaptive Moment Estimation (Adam) and RMSProp optimisation techniques for wildlife animal classification using convolutional neural networks," *Revue d'Intelligence Artificielle*, vol. 37, no. 4, pp. 1023–1030, 2023. doi: 10.18280/ria.370424

[8] M. Reyad, A. M. Sarhan, and M. Arafa, "A modified Adam algorithm for deep neural network optimization," *Neural Comput. Appl.*, vol. 35, no. 23, pp. 17095–17112, 2023. doi: 10.1007/s00521-023-08568-z

[9] R. Arablouei, L. Wang, L. Currie, J. Yates, F. A. P. Alvarenga, and G. J. Bishop-Hurley, "Animal behavior classification via deep learning on embedded systems," *Comput. Electron. Agric*., vol. 207, 107707, 2023. doi: 10.1016/j.compag.2023.107707

[10] T. N. Doan, "A novel real-time insect detection system on mobile smart devices," *Journal of Advances in Information Technology*, vol. 16, no. 5, pp. 655–665, 2025. doi: 10.12720/jait.16.5.655-665
[11] S. Binta-Islam, D. Valles, T. J. Hibbitts, W. A. Ryberg, D. K. Walkup, and M. R. J. Forstner, "Animal species recognition with deep convolutional neural networks from ecological camera trap images," *Animals*, vol. 13, no. 9, 1256, 2023. doi: 10.3390/ani13091526
[12] G. Moallem, D. D. Pathirage, J. Reznick, J. Gallagher, H. Sari-Sarraf, "An explainable deep vision system for animal classification and detection in trail-camera images with automatic post-deployment retraining," *Knowledge-Based Systems*, vol. 216, 106815, 2021. https://doi.org/10.1016/j.knosys.2021.106815
[13] P. Sykora, P. Kamencay, R. Hlavata, and R. Hudec, "Overview and comparison of deep neural networks for wildlife recognition using infrared images," *AI*, vol. 5, no. 4, pp. 2801–2828, 2024. doi: 10.3390/ai5040135
[14] E. M. Dogo, O. J. Afolabi, and B. Twala, "On the relative impact of optimizers on convolutional neural networks with varying depth and width for image classification," *Applied Sciences*, vol. 12, no. 23, 11976, 2022. doi: 10.3390/app122311976
[15] W. Luo, G. Zhang, Q. Shao, Y. Zhao, D. Wang, X, Zhang, K. Liu, X. Li, J. Liu, and P. Wang, "An efficient visual servo tracker for herd and monitoring by UAV," *Scientific Reports*, vol. 14, no 1, 10463, 2024. https://doi.org/10.1038/s41598-024-60445-4
[16] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951. https://doi.org/10.1214/aoms/1177729586
[17] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT'2010: 19th International Conference on Computational Statistics*, 2010, pp. 177–186. https://doi.org/10.1007/978-3-7908-2604-3_16
[18] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964. https://doi.org/10.1016/0041-5553(64)90137-5
[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[20] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. International Conference on Machine Learning*, 2013, pp. 1139–1147.
[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res*., vol. 12, no. 7, 2011.
[22] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint, arXiv:1609.04747, 2016.
[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint, arXiv:1412.6980, 2014,
[24] A. Burlacu, C. Cosmin, A. Panainte, C. Pascal, and C. Lazar, "Realtime image image-based visual servoing architecture for manipulator robots," in *Proc. International Conference on Computer Vision Theory and Application*, 2011, pp. 502–510. doi: 10.5220/0003367005020510
[25] A. Swanson, M. Kosmala, C. Lintott *et al*., "Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna," *Scientific Data*, vol. 2, no. 1, pp. 1–14, 2015. doi:10.5061/dryad.5pt92