AUTOMATED IDENTIFICATION OF NON-INFORMATIVE IMAGES IN CAMERA TRAP DATA USING DEEP LEARNING APPROACHES

Sangam Babu Neupane[1]*, Sandhya Sharma[1], Bishnu Prasad Gautam[2], Kazuhiko Sato[1]

[1] Muroran Institute of Technology, 27-1 Mizumoto-cho, Muroran, Hokkaido 050-8585, Japan

[2] Kanazawa Gakuen University, 10 Suemachi, Kanazawa, Ishikawa 920-1392, Japan

Abstract

Camera traps are used primarily for the monitoring, conservation, and habitat management of wildlife, especially rare and endangered species. They are preferred among ecological community members who are engaged in protecting and studying the animal ecosystem. Camera traps generate a vast number of images over a given period; however, the majority of these images do not contain any animal sightings. These non-informative images, which are referred to as "garbage images" or "empty images," are irrelevant and should be eliminated during the research process. The manual process of separating these non-relevant images from those containing the desired animal sightings consumes a significant amount of time and resources, which hinders the efficiency of the research process. To address this challenge, we compiled a diverse collection of camera trap images from various locations and created a dataset consisting of 38,669 empty and 35,083 non-empty images. Additionally, we developed dataset filtering software that sorts the images into folders designated for empty and non-empty images. This software enabled the manual selection and classification of images and facilitated the creation of a large, labeled dataset. Evaluating various state-of-the-art deep learning algorithms using our custom dataset revealed that transfer learning with EfficientNet_B2 resulted in greater than 98% accuracy in distinguishing between empty and non-empty images. With these results, we have been developing an automated filtering program that utilizes deep learning models to effectively discard empty images from camera trap datasets.

Keywords: *Camera Traps, Deep Learning, Transfer Learning, Image Classification, Automated Filtering, Dataset Filtering Software, Image Processing*

1. INTRODUCTION

Animals play a crucial role in the natural ecosystem. The conservation of animals for the continuity of the ecosystem is an important topic, thus ecologists and conservation biologists all over the world are researching possible ways to maintain balance in the ecosystem through different means [1]-[2]. Technology is being used to learn more about ecological and animal behavior. Unmanned Aerial Vehicles, camera traps, and Global Positioning Systems are among the most well-known and widely used technologies in wildlife monitoring [3]. Advancements in deep learning technologies and the availability of natural images from actual monitoring sites have introduced the possibility of using computer vision to observe animals. However, camera traps, which are some of the most-used monitoring devices in wildlife studies [4], have limitations that make the research process time-consuming.

Camera traps are devices with a motion sensor attached to them, and they capture an image whenever movement occurs in the range of the sensor [5]. Capturing and saving every image that is triggered by different natural phenomena like wind, broken tree branches, and other motion around the device fills the available memory with empty images "images without any animal sightings". In the African savanna, 225 cameras were deployed over an area of 1,125 km^2 in Serengeti National Park, resulting in the collection of 3.2 million images. Of these images, a staggering 75% were found to be empty images, or images without any animal sightings [6]. The collected garbage images need to be discarded during the analysis of the image data from the camera trap. The manual process of sorting through these non-informative images and discarding them is time-consuming and resource-intensive, and it hinders the efficiency of the research process. In light of these challenges, developing effective methods for

separating meaningful images from non-informative images is crucial. This would not only save time and resources, but would also enhance the accuracy of wildlife monitoring and ultimately contribute to the success of conservation and habitat management efforts.

With the advent of deep learning, which is a subset of machine learning, there is potential for a more automated and efficient solution. Deep learning algorithms can extract hierarchical features from images and categorize them into specific output classes, which enables faster and more accurate analysis of large datasets [7]. However, the performance of deep learning algorithms can be impacted by factors such as the quality of the input data, the size and diversity of the training datasets, and the complexity of the target classes. Additionally, when the algorithms are applied to natural datasets like camera trap data, they have difficulty accurately identifying camouflaged or obscure animals in a wild environment. Deep learning has the potential to revolutionize the way camera trap data is analyzed and has the potential to greatly enhance our understanding of wildlife populations and behavior. This study demonstrated the efficacy of utilizing transfer learning [8], a common technique in deep learning, to enhance the accuracy of camera trap data analysis. By leveraging models pre-trained on large datasets, we were able to attain superior results compared with traditional deep learning architecture and data augmentation/regularization methods. Transfer learning allowed the model to utilize the knowledge acquired from previous training on a related task, thus improving its performance on the new task. This has the potential to advance our understanding of wildlife populations and behaviors through a more precise analysis of camera trap data.

Recently, the application of deep learning to identify animals captured in camera trap images has garnered attention among researchers [10],[11]. Despite numerous studies done to evaluate the performance of different deep learning architectures [11], the relationship between model performance and inference latency has yet to be fully explored when these models are deployed in real-world automatic identification systems. Transfer learning from high-performance models facilitates the categorization of large datasets, but this approach often results in high latency and slower recognition speeds, particularly for datasets that contain small animals or images captured under nighttime conditions when the animals are more difficult to detect.

Our research focuses on improving the processing of camera trap data by addressing two major issues in the deployment of deep learning models directly on edge devices [12],[13]:

1. Latency: The direct deployment of models on devices like Raspberry Pi results in slow image recognition, which hinders the real-time processing with high accuracy for large datasets.
2. Cost and Inefficiency: Replacing the numerous existing camera traps with new devices to accommodate the deployment of deep learning models would be an expensive and inefficient solution.

Instead, we aim to tackle the challenge of efficiently identifying and extracting relevant images from the existing camera trap data stores using an automatic identification system. By developing a model that could accurately categorize images with minimal latency, it would be possible to tackle the existing problems and enhance the effectiveness of identifying wildlife in their natural habitats. This optimized model could then be integrated into edge devices to bring the solution closer to real-world implementation.

2. MATERIALS AND METHODS

Our research draws data primarily on two significant datasets: the Caltech Camera Traps [14] and the GigaDB repository [15]. These datasets are combined to form a unified dataset of empty and non-empty images, with the aim of increasing dataset diversity and reducing the risk of overfitting. Camera traps capture images when they are triggered by moving objects such as waving vegetation or moving animals, and they produce a set of images for each capture event. The resulting dataset encompasses a wide range of environmental conditions, including day, night, sunny, and more.

2.1. Datasets

The dataset is a crucial component of any machine learning or deep learning project, and our research aims to employ the supervised classification method to automatically identify empty and non-empty images. However, we encountered a challenge in locating a suitable dataset that contained labeled and classified empty and non-empty images to train the new model. To address this issue, we compiled a dataset by combining camera trap images from two camera trap sites, and we gradually expanded it over time. We selected these datasets based on their diversity of animal species, camera placement, and environmental conditions, which pose distinct challenges for developing robust machine learning models.

Caltech Camera Traps [14]: The Caltech Camera Traps dataset is a comprehensive dataset of 243,100 images that were acquired from 140 distinct camera locations in the Southwestern United States.

These images comprise 21 different animal categories, which are primarily classified at the species level (e.g., opossum, raccoon, and coyote). This dataset is particularly relevant for machine learning and computer vision research, given its large size and the diversity of environments in which the images were captured (such as day, night, and sunny conditions) (Figure 1). Among the 66,000 images in the dataset, a significant proportion (70%) of the images were found to be empty.

For our experiments, we select a subset from the unlabeled images, and we increment the dataset gradually to improve the accuracy. Although the dataset is initially presented in the object detection format of the Common Objects in Context dataset, we require labeled data for the binary classification to the empty and non-empty classes. To address this requirement, we manually create a dataset with the empty and non-empty classes and further divide it into a training set and a testing set in a random manner in the proportion of 8:2.

GigaDB Dataset [15]: The Carrizo Plain National Monument study involved the deployment of 20 motion sensor trail cameras at three locations along Elkhorn Road within the Carrizo Plain National Monument. The study was intended to examine the interactions between the animals and the dominant shrub species, Ephedra californica, in the San Joaquin Desert ecosystem. The cameras were placed in both shrub understory and open microhabitats at ground level. They were operational from April 1 to July 5, 2015. During this time, over 420,000 images were captured. Of these images, 100,000 were reviewed for image quality, the presence of animals, date and temperature stamps, and data integrity. The data were then organized into 467 individual compressed directories and uploaded to GigaDB, where they were labeled according to the date, microsite type, site ID, shrub ID, and camera ID. The camera trap event data are also available in GigaDB, along with the sampling metadata table, which includes accurate dates for the study.

Fig. 1. Screenshot of unlabeled images from Caltech camera trap dataset.

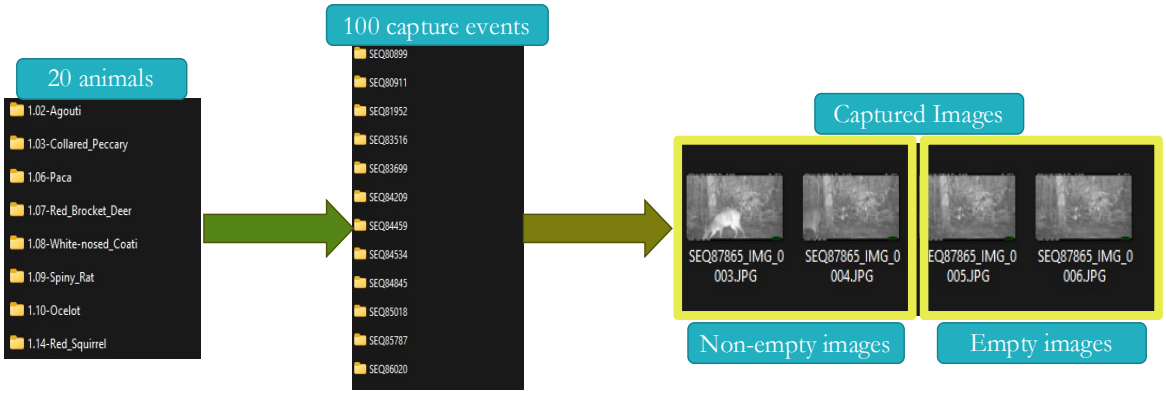

Fig. 2. Process of image selection from GigaDB dataset.

For our experiment, we utilize the data from site 1, in which 20 animals were classified and organized in labeled folders. Each of the folders, which were denoted by the animal's name, contained between 87 to 100 capture events that consisted of images captured by the trail cameras (Figure 2). It should be noted that utilizing data from only one site may result in a biased outcome during the animal classification process, as the images may possess similar properties. However, the presence of 20 distinct animal species captured against diverse backgrounds minimizes this concern, especially because the GigaDB data is merged with the Caltech dataset.

2.1.1. Dataset creation app

In our study, the task of categorizing images based on the presence of animals in the frame is a manual process that involves the inspection of each image to determine the presence of animals and assign the image to either the "empty folder" if no animals are present, or to the "non-empty folder" if an animal is present. The manual nature of this process, particularly the need to view the images in a photo viewer and then to move them to the appropriate folder, makes dataset creation a challenging and time-consuming task.

To optimize the manual categorization, a graphical user interface (GUI) application was developed using PySimpleGUI. The application serves as an image viewer and offers functionality for categorizing images as either empty or non-empty based on the presence of animals in the frame. The GUI includes navigation buttons such as "next" and "previous" to traverse through the images, and a "delete" button to eliminate outliers prior to training the binary classifier for image classification (Figure 3). This tool facilitates a more efficient and less time-consuming process for the categorization of the images.

2.1.2. Custom dataset

The construction of a new custom dataset for the classification of empty and non-empty images was achieved by merging images from both the Caltech and the GigaDB datasets and filtering them using our dataset creation tool. The resulting dataset comprises a mixture of empty and non-empty images, and 63.46% of the images in the combined dataset are identified as empty or non-informative. To ensure an accurate and unbiased assessment of the performance of the classification model, it is crucial for the dataset to have a balanced distribution of classes [16]. Training on an imbalanced dataset can result in a biased evaluation of a model that favors the class with more data and ignores the classes with fewer samples [17]. This can also lead to overfitting on the empty class and can result in a high rate of misclassification of informative images as non-informative. A common approach to balancing datasets is to duplicate the images, but this can lead to overfitting, as the same images are used repeatedly [18]. To overcome this issue, we employ data augmentation techniques to enhance the robustness of the model [19]. These techniques involve altering the images through rotation, scaling, flipping, cropping, and color distortions while preserving the salient features of the original images. The new dataset that is created using these techniques is then used for the training and testing of the image classification model.

The combination of the Caltech and GigaDB datasets provides a more diverse and robust dataset and enables the classification model to better identify the presence of animals in the images.

Fig. 3. Manual dataset creation app. a) Workflow of manual dataset creation app. b) GUI of manual animal dataset creation app based on PySimpleGUI.

2.2. Model Implementation

In our research, we utilize PyTorch [20], an open-source machine learning and deep learning framework, for its exceptional flexibility and compatibility with the Python programming language. The image inputs are transformed into tensors, which serve as the fundamental data structure in deep learning [21]. These tensors are then used to train our deep learning model, and they allow the model to learn the patterns and relationships within the data. To achieve the best accuracy on our custom binary classification dataset, we experiment with several different models, including the TinyVGG architecture [22] and Vision Transformer [23], and we use transfer learning with EfficientNet [24] models. In our process, we first preprocess the dataset, and then we build and optimize our model. The model is then fitted and evaluated using several experimental steps, with the goal of achieving a desired accuracy of greater than 97% on our dataset. Once the desired accuracy is achieved, the model is saved and deployed in the Gradio app for single image classification experiments. If the model performs well with low latency, it is then deployed into our automatic categorization app.

It is important to note that transfer learning is a crucial aspect of our research. This approach allows us to leverage the knowledge obtained by deep learning models that are pre-trained on large datasets and apply it to our custom dataset. This significantly reduces the amount of time and computational resources needed to train the model and still yields accurate results. These techniques allow us to effectively process and classify image data, as well as to model the sequential relationships between image frames. Figure 4 shows the workflow of our experimental procedure.

TinyVGG [22]**:** In this study, we study the potential improvement in a visual data classification task using a Convolutional Neural Network (CNN) model. The first model adopted is TinyVGG, a typical CNN structure consisting of input, convolutional, activation, pooling, and output layers. The model is trained to minimize the loss function by updating its internal parameters (weights and biases) through optimization. The best hyperparameter values, which can impact the performance of the model, are determined through experimentation or by referring to similar problems. The goal of a CNN is to

compress the information in high-dimensional space to a lower-dimensional space by learning predictable patterns in the data. We employ color images in our study instead of grayscale images for the input. To speed up the computation, operator fusion is utilized in the forward method. A custom image is preprocessed and converted to a tensor to match the type of data used in training.

EfficientNet [24]: Transfer learning is a technique in which a model that is pre-trained, usually on a large dataset, is fine-tuned on a smaller dataset for a specific task, which enables it to benefit from the generic features learned during pre-training and to be fine-tuned on task-specific features. In this study, we experiment with three transfer learning models, and we utilize pre-trained models from torchvision.models[25] for image classification. The models we use are EfficientNet_B0, EfficientNet_B2, and EfficientNetV2_S, all with IMAGENET1K_V1[26] weights. The custom data for the models requires images that are normalized in the same way as the original training data, as pre-trained models expect input images to be 3-channel RGB images having the shape (3 x H x W), where H and W are at least 224. The images must be loaded in the range of [0, 1] and then normalized with mean = [0.485, 0.456, 0.456] and std = [0.229, 0.224, 0.225]. EfficientNet_B0 can achieve an accuracy of ~77.7% across ImageNet's 1000 classes[27]. The model is composed of three parts: features, avgpool, and classifier. The features part consists of convolutional layers and activation layers that learn the base representation of the image data; avgpool takes the average of the output of the features layer(s) and transforms it into a feature vector; and classifier transforms the feature vector into a vector with the same dimensionality as the number of output classes.

Vision Transformer [24]: We also experimented using Vision Transformer (ViT16) architecture with PyTorch to work on our image classification problem. The Transformer architecture is a neural network architecture that uses an attention mechanism as its main learning layer. We focus on replicating the original vanilla Vision Transformer. To build the ViT architecture, we create a class called ViT that inherits from torch.nn.Module. We initialize this class with the hyperparameters from the original ViT architecture, and ensure that the image size is divisible by the patch size. We calculate the number of patches, create a learnable class and position embedding token, set up the embedding dropout layer, and create the patch embedding layer and a series of Transformer Encoder blocks. We also create an MLP head as the classifier. In the forward method, we obtain the batch size of the input, create the patch and class token embedding, and add the position embedding. We then pass it through the dropout layer, the stack of Transformer Encoder layers, and the classifier head.

3. EXPERIMENTS

This study has as its primary objective to evaluate the efficacy of various deep learning models to facilitate the integration of the models into an automated system for categorizing images into empty and non-empty classes. To this end, we conduct experiments on different image datasets collected from two distinct geographical locations. This section provides a detailed description of the experimental setup, including the experimental scenario and computer specifications used in the study. Our experiments involve the integration of deep learning models on a computer system to achieve the research objectives.

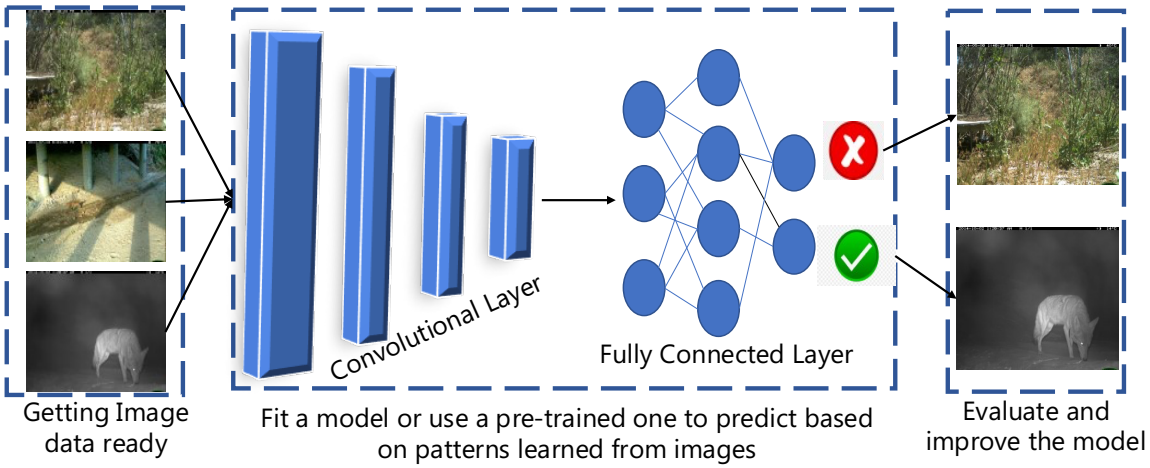

Fig. 4. Overview of the experimental procedure.

3.1. Experimental Setup

The pseudo code for the algorithmic flow of our experimental setup is presented in Algorithm 1. In a supervised learning scenario, we need a large dataset of labeled images and since the data we needed were random images of empty or non-empty images. We gradually increase the dataset, starting with a small dataset and increasing the dataset as we continue to experiment and evaluate the results. The experiments use the different state-of-the art architectures available. We use the architectures and models that suit our problem of identifying the empty and non-empty images. TensorBoard[28], which is an evaluation framework used for tracking a large numbers of experiments, is used to visualize the result of the architecture with different hyperparameter tunings. We train our model and track every model using TensorBoard. We follow the procedure of simultaneously tracking the experiments with different models and different parameters instead of testing them one by one. Experiment tracking is done to evaluate model performance in different circumstances to find the best-performing model. We then make predictions using the best-performing model to determine the usability of our model in automated image categorization applications. Overfitting and underfitting of the dataset are observed and further experiments are done to make the model ideal for testing with unknown samples in the test dataset.

Algorithm 1 Experimental Setup

1. **Start from small dataset.**
2. **Customize model to fit the problem of finding informative images.**
3. **while Result is not satisfactory do:**
4. **Increase dataset with labels using dataset creation app.**
5. **Augment the data.**
6. **Train the model and track results**
7. **Visualize in TensorBoard**
8. **if accuracy of model >= 97% and model ≠ overfitting then:**
9. **Save the best model and make predictions.**
10. **end if**
11. **end while**

Table 1. Summary of experiments: model architecture, epochs, batch size, total parameters, trainable parameters, model size, and training time

Experiment Number	Model Architecture	Epochs	Batch size	Total Parameters	Trainable Parameters	Model Size (MB)	Time (h)
1	TinyVGG	5	8	65,732	65,732	0.25	0.969
2	EfficientNet_B0	5	8	4,010,110	2,562	15.58	0.975
3	EfficientNet_B2	5	8	7,703,812	2,818	29.82	0.975
4	EfficientNetV2_S	5	8	20,180,050	2,562	77.85	0.976
5	ViT	5	8	85,800,194	1,538	327.36	1.225
6	TinyVGG	10	32	65,732	65,732	0.26	2.255
7	EfficientNet_B0	10	32	4,010,110	2,562	15.96	2.282
8	EfficientNet_B2	10	32	7,703,812	2,818	30.54	2.296
9	EfficientNetV2_S	10	32	20,180,050	2,562	79.72	2.292
10	ViT	10	32	85,800,194	1,538	327.36	2.672

3.2. Experiments Scenario

To identify the optimal deep learning model for our image categorization application, ten experiments were conducted to assess the effectiveness of different models in image classification tasks. A comprehensive summary of these experiments can be found in Table 1.The experiments involve the manipulation of critical parameters, such as model architecture, number of epochs, batch size, number of total parameters, number of trainable parameters, model size, and training time. The experiments were carried out with the TinyVGG, EfficientNet_B0, EfficientNet_B2, EfficientNetV2_S, and ViT models. Each experiment was executed for 5 epochs or 10 epochs, with batch sizes of 8 and 32 and a learning rate of 0.001 (scientific notation: 1e-3). The table presents important insights into the total number of parameters and trainable parameters for each model, which are essential factors in determining the model's ability to learn from the training data. Additionally, the model size (in MB) and training time (in hours) are also provided. These critical metrics are necessary for selecting an appropriate model, as a larger model may need more time and computational resources to train. The capacity of a model is often correlated with its number of parameters, which can increase the model's ability to learn from data. However, this does not guarantee that the model will utilize this additional capacity. The ViT model has a large number of parameters, indicating a high capacity for learning if given sufficient data. However, this also leads to larger model file sizes and longer inference times. The table provides a thorough overview of the experiments conducted, facilitating an objective comparison of the performance of the different model architectures.

3.3. PC Specification

This study was conducted using a computer with the following specifications: Intel® Core™ i7-9700 CPU @ 3.00 GHz, 16.0 GB RAM, and an NVIDIA Geforce GTX 1660 Ti GPU. The computer was employed to train and evaluate machine learning models for the study. The specifications of the computer were meticulously selected to ensure the optimal conditions for conducting the experiments. The high processing power of the Intel i7-9700 CPU, and the 16 GB of installed RAM enabled the large datasets to be processed efficiently and prevented potential bottlenecks during the experiment. The NVIDIA Geforce GTX 1660 Ti GPU, with its high-performance computing capabilities, was particularly useful in accelerating the training of the deep learning models. These specifications were carefully considered to ensure that the experiments were reliable and produced accurate results. With the given PC specifications, decent results were obtained, suggesting that transfer learning with

parameters kept constant is suitable for the task of classification of real-world data. However, it was observed that the ViT model had a large number of parameters and a large size, and it required more time for execution.

4. RESULTS

In our study, we aimed to determine the best model and the appropriate hyperparameters for achieving high accuracy in classifying empty and non-empty image data. Given that real-world data is often highly imbalanced, with some classes having very few samples, it is crucial to select a model and parameters that can perform well on such datasets. As our model is intended for deployment in a PC application, factors such as latency and model size must be considered. We conducted experiments on our custom dataset using various experimental scenarios, considering the detection criteria for real-world data. We calculated the recall, precision, and F1 score for all models, in addition to the overall accuracy.

The present study is intended to evaluate the performance of five different model architectures for an automated categorization application that uses image data. The models under consideration include three transfer learning models whose CNN architectures use EfficientNet, one model that uses the TinyVGG architecture, and one model that uses transfer learning with the transformer-based model ViT16.

The models were subjected to a variety of experimental scenarios to determine their effectiveness and suitability for the categorization application. The results of the experiment show that the ViT16 model performed best, followed by the EfficientnetB2 model. However, the TinyVGG architecture showed overfitting and performed poorly compared with the other transfer learning models (Figure 5). These findings are essential to selecting an appropriate model architecture for an automated categorization application that uses image data.

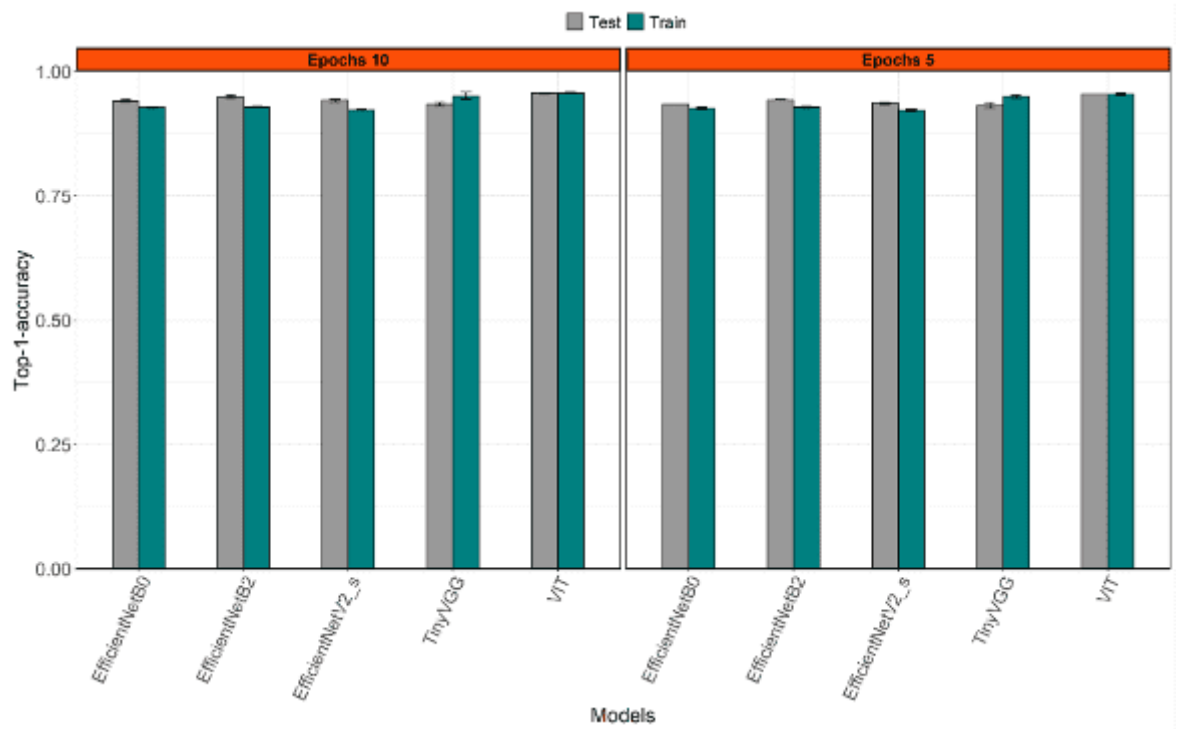

Fig. 5. Top-1 accuracy comparison of five different model architectures on our custom dataset in a variety of experimental setups.

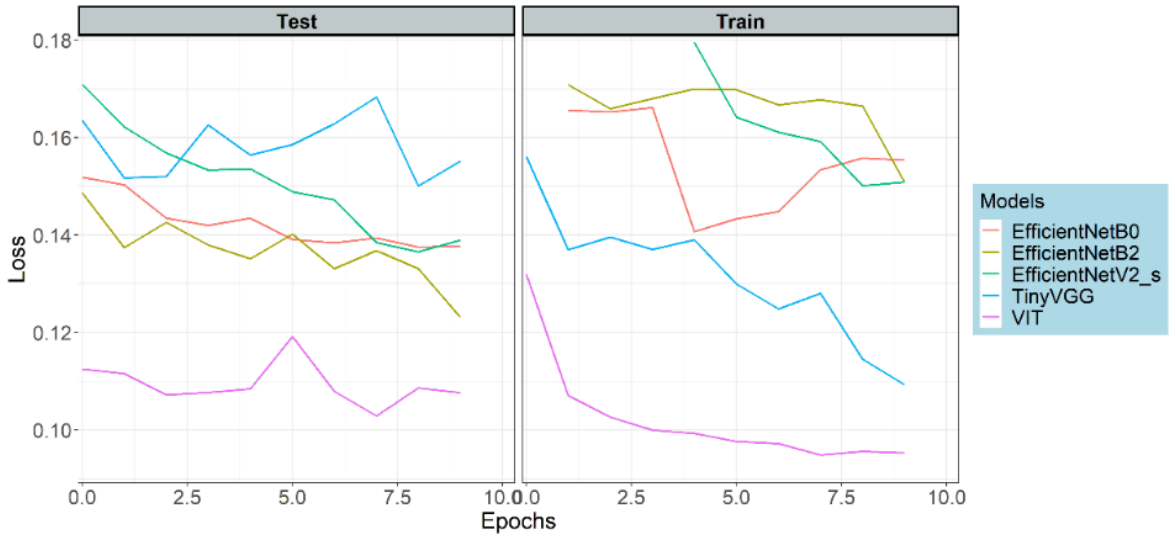

Fig. 6. Testing and training loss values plotted for different models over 10 epochs.

To evaluate the performance of various model architectures on our custom dataset, we calculated and plotted the loss values for five different experiments in Figure 6. The model with the lowest and smoothest loss was determined to be the Vision Transformer-based model, which minimized the loss function better than the other models. However, a closer inspection of the graph reveals that some minor underfitting may be present in the ViT model. By comparison, the EfficientNet B2 model showed smaller loss values as expected, indicating that it performed well on our dataset. In contrast, the TinyVGG model had an overfitting issue, as can be observed by the loss curve. Figure 6 shows the graph of the loss curve on the testing and training dataset only for the experiments conducted with 10 epochs. This was because the loss curves for experiments performed with 5 epochs and a batch size of 8 were not significantly different from those performed with 10 epochs and a batch size of 32. Additionally, we chose to present the loss curve for the 10-epoch experiments, because the main categorization application will be based on the results obtained from experiments that run up to 10 epochs.

Table 2. Model performance evaluation for each model.

Model	classes	TP_rate	FP_rate	Recall	Precision	F1 score
TinyVGG	empty	0.93	0.037	0.998	0.962	0.98
	non-empty	0.95	0.034	0.997	0.966	0.981
EfficientNet_B0	empty	0.933	0.03	0.991	0.968	0.979
	non-empty	0.972	0.017	0.999	0.983	0.991
EfficientNet_B2	empty	0.945	0.029	0.994	0.97	0.982
	non-empty	0.995	0.004	1	0.996	0.998
EfficientNetV2_S	empty	0.946	0.032	1	0.968	0.983
	non-empty	0.98	0.014	0.999	0.986	0.992
ViT	empty	0.948	0.033	0.999	0.966	0.982
	non-empty	0.996	0.003	1	0.997	0.998

Table 2 presents the evaluation metrics for five different models that were trained on a custom dataset of empty and non-empty images. The models include TinyVGG, EfficientNet_B0, EfficientNet_B2, EfficientNetV2_S, and Vision Transformer (denoted as ViT). The evaluation metrics include true positive rate (TP rate), false positive rate (FP rate), recall, precision, and F1 score for each class (empty

and non-empty). Overall, the models perform well in classifying the images, with high TP rates and low FP rates. The EfficientNet_B2 and ViT models achieve the highest performance, and have TP rates of 0.945 and 0.948, respectively, for the empty class, and TP rates of 0.995 and 0.996, respectively, for the non-empty class. These same models also achieve perfect precision and high F1 scores, indicating that they are able to accurately identify both empty and non-empty images. The TinyVGG model, however, has a lower TP rate and higher FP rate than do the other models, particularly for the empty class, indicating that it has difficulty in accurately classifying these images. Additionally, this model is overfitted, as shown by the loss curve analysis, which may contribute to its lower performance.

Figure 7 presents the performance evaluation of five different models on a dataset that consists of empty and non-empty images. The evaluation metrics used in this study include the weighted average F1 score, macro-averaged F1 score, and micro average (accuracy). Among the models evaluated, EfficientNet_B2 demonstrates the best performance and has the highest weighted average F1 score of 0.9898 and the highest macro-averaged F1 score of 0.9901. This suggests that EfficientNet_B2 achieves the best balance between precision and recall in detecting both empty and non-empty images in the dataset. Additionally, the model achieves a high micro average (accuracy) of 0.9828, which indicates that it performs well in overall classification accuracy. However, TinyVGG demonstrates the lowest performance among the evaluated models. It has the lowest weighted average F1 score of 0.9805 and the lowest macro-averaged F1 score of 0.9805, which implies that the model has lower precision and recall in classifying empty and non-empty images. This model also achieves a lower micro average (accuracy) of 0.9638 than the other evaluated models. Overall, the results suggest that EfficientNet_B2 is the most effective model for classifying empty and non-empty images, whereas TinyVGG is the least effective for this task. These findings may have important implications for applications other than wildlife management, such as waste management and environmental monitoring, where accurate identification of empty and non-empty images can provide valuable insight for decision-making.

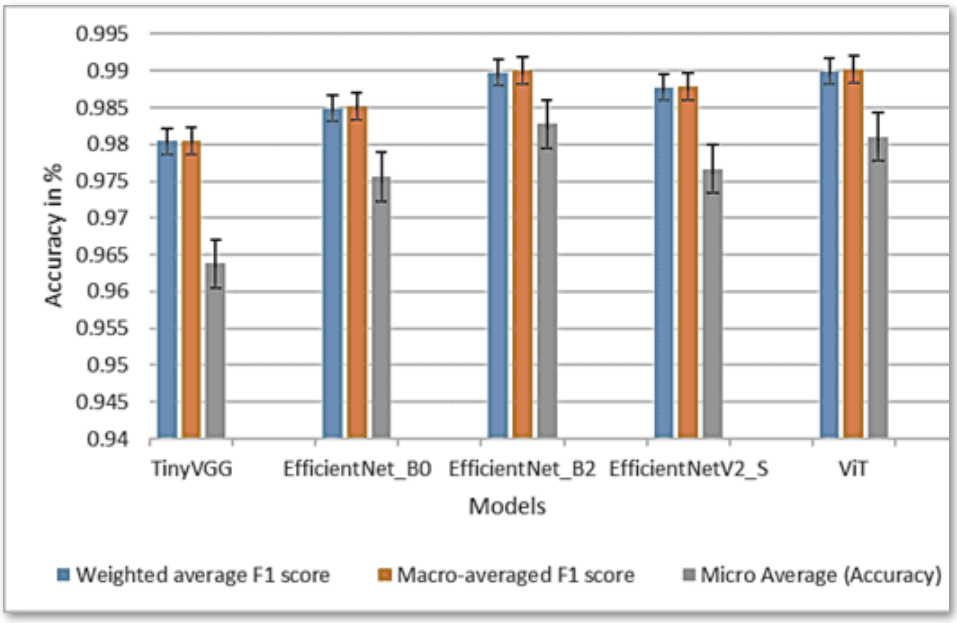

Fig. 7. Bar graph for Weighted average F1 score, Macro-averaged F1 score, and accuracy with standard error.

5. DISCUSSION

An efficient analysis of camera trap images is essential for categorizing non-informative images from the large volume of data acquired. To achieve this, we compared five different models for the binary classification of an image dataset, and we combined two different datasets to incorporate varied backgrounds in the detection of animals in the image frame. We tested all the models on totally unknown

images after hyper-tuning them with different parameters. Our experiments uncovered overfitting and underfitting issues in some of the models, which could be attributed to the similarity of the background to items that resemble animals, such as plastic bags and rocks. Our experimental results showed that the EfficientNet_B2 model achieved the highest scores for all three metrics, with a weighted average F1 score of 0.98, a macro-averaged F1 score of 0.99, and a micro average accuracy of 0.98. These scores determined it to be the most effective model for non-informative image classification on our dataset. Conversely, TinyVGG achieved the lowest scores for all three metrics, indicating its ineffectiveness for classification on the dataset. The reason that the EfficientNet_B2 model performed better than the TinyVGG may have been its deeper and more complex architecture, which enabled it to capture more detailed features in the image [26]. Moreover, the EfficientNet_B2 model was fine-tuned with a larger set of hyperparameters, which may have enabled it to more accurately classify the non-informative images in the dataset. It is also worth noting that the performance of the models varied according to the class of images being analyzed. All the models achieved high accuracy in detecting empty images, with recall values above 0.99 and precision values above 0.96. However, in detecting non-empty images, there was greater variation in the performance of the models, with EfficientNet_B2 and ViT achieving the highest F1 scores and TinyVGG achieving the lowest F1 score. This suggests that the features that are important for detecting non-empty images could be more complex and difficult to capture than are the features that are important for detecting empty images. The images in our dataset present a wide range of scenarios that present a challenge to accurately classifying non-informative images. For example, the dataset includes night images, images with clear and visible animals, and images where only parts of the animal are visible, such as limbs or other body parts. Additionally, there are images with poor illumination, overexposed images taken in bright sunlight, complex animal poses, or images in which animals are running. Some images in the dataset may also be blurred, while others depict animals hiding behind bushes or positioned far from the camera, which makes it difficult to identify them. Furthermore, some images only show the eyes of animals; these are captured exclusively at night. The different scenarios and variations that are present in the images may have an impact on the classification performance of the models [27]. Images with poor illumination, overexposed images, and blurred images could have low contrast and unclear details, which would make it difficult for the models to distinguish between the target objects and the background elements. Complex animal poses, sudden unanticipated images, and animals in a bush or behind a bush can also add to the difficulty of classification, as bushes may obscure the view of the animal or make it difficult to identify its features accurately. Additionally, images in which animals are positioned far from the camera and are difficult to identify, or images where only the eyes of the animal are visible at night, may require a more advanced classification algorithm or feature extraction techniques to accurately classify the image. Overall, these factors affect the accuracy and performance of the models in classifying non-informative images.

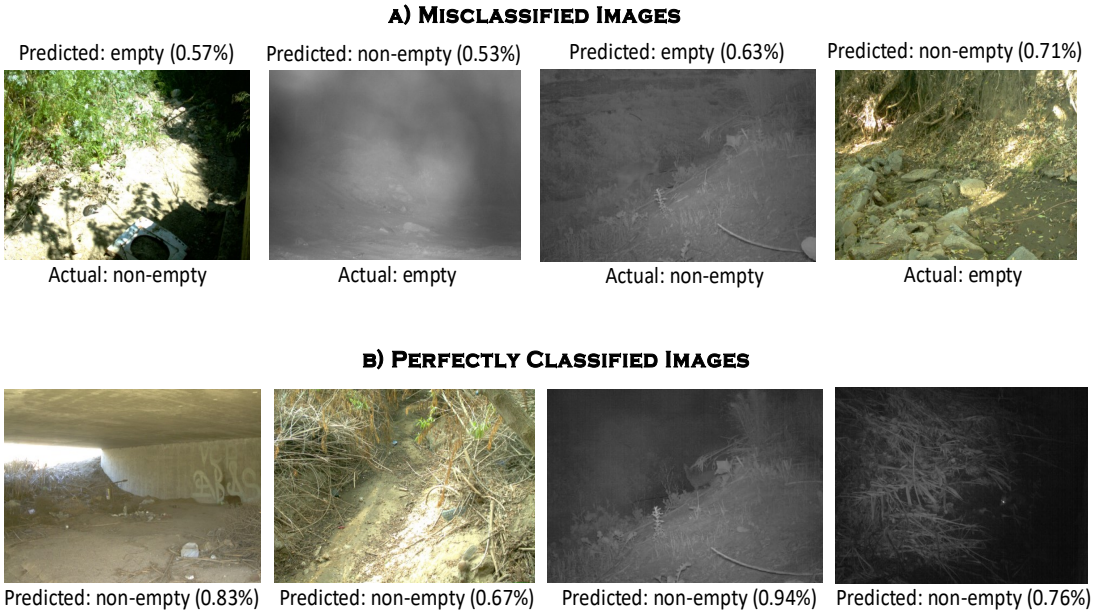

Fig. 8. Examples of a) misclassified images and b) perfectly classified images.

In Figure 8a, we present some of the images that were misclassified by our model during testing. Our observations show that images that contain animals that are camouflaged due to sunlight or blending with the background, images that contain too many small objects such as scattered rocks that resemble animal species, and blurred images were classified as non-empty images. Although these images were misclassified, the low prediction accuracy suggests that the model was confused by the information presented in the image. Figure 8b, on the other hand, displays images that are difficult for the human eye to classify but that were classified perfectly by the model. Even animals that were far away from the camera were accurately classified, as were animals with minimal distinguishing features. The high accuracy of non-empty image classification indicates that the model is reliable for automated image categorization despite its minor errors in classifying images with animals as empty images.

There are several limitations to our research that should be noted. Firstly, our study utilized only images from datasets that originated from two specific locations, which may limit the generalizability of our findings to other datasets. Additionally, the dataset used in this study contained a limited number of animal species, and these species may not be representative of all species that could be encountered in camera trap images. Furthermore, our study focused on the binary classification of images into “empty” and “non-empty” categories and did not explore other, more complex classification tasks, such as identifying specific animal species or counting the number of animals in an image. A further limitation is that the models were trained and tested on only two types of background variation (day and night). It is possible that other types of backgrounds or lighting conditions could affect the performance of the models differently. Lastly, the study did not consider the effect of different camera trap settings, such as the placement and trigger settings, which could affect the quality of the images and thus the performance of the classification models. Therefore, although our findings suggest that the EfficientNet_B2 model is effective for non-informative image classification on this dataset, further research is needed to confirm our findings on different datasets and under different conditions, and to explore classification tasks that are more complex than binary classification.

Our study on non-informative image classification using different models has significant implications for the practice of wildlife conservation and management. Camera traps are widely used to monitor and study wildlife populations, and the efficient classification of images is essential to the effective management of these populations. Our findings suggest that the EfficientNet_B2 model is the most effective model for non-informative image classification on our dataset; this finding could lead to the development of more efficient and accurate image classification systems for camera trap data.

Furthermore, our study highlights the challenges and limitations of using machine learning models for image classification, especially with images that contain complex scenarios such as poor illumination and animal camouflage, and with blurry images. These challenges must be addressed to ensure the accuracy and reliability of an automated image classification system. The findings of our study could also be used to inform policy decisions about wildlife management and conservation. Automated image classification systems have the potential to save significant time and resources, which can be used to develop more effective conservation and management strategies. Moreover, these systems can help detect and monitor endangered and rare species and can aid in their protection and conservation. Our study contributes to the growing body of research on machine learning-based image classification for wildlife management and conservation, and our findings have important practical implications for the development of more efficient and accurate image classification systems for camera trap data.

6. CONCLUSIONS

Based on the experiments and analysis conducted in this study, we conclude that the EfficientNet_B2 model is the most effective model for non-informative image classification on our dataset. This model achieved high accuracy, recall, and precision values for both empty and non-empty images, which is an indication of its reliability for automated image categorization despite some minor errors in classifying images with animals as empty images. Additionally, we observed that the performance of all models varied according to the class of images being analyzed, with some models performing better than others for specific image types. The results of this study have important implications for the field of wildlife conservation and monitoring, as the automated filtering of garbage images from camera trap data can significantly increase the efficiency of researchers, and reliable automated image categorization greatly improves the efficiency and accuracy of camera trap image analysis. However, it is important to note the limitations of our study, such as our use of a small dataset and the limited number of models we tested. Future research could expand on our findings by testing additional models and datasets and by creating an application that can automatically categorize images into particular folders with high accuracy, thereby reducing the workload for ecology researchers and wildlife conservatives.

ACKNOWLEDGMENTS

We gratefully acknowledge the support provided by the Project base AI (PBL-AI) program and the Ministry of Education, Culture, Sports, Science, and Technology (Monbukagakusho: MEXT) scholarship awarded to Sandhya Sharma. We also thank NTT-Docomo for providing the International Student Scholarship to Sangam Babu Neupane. We would like to express our sincere appreciation to Enago (www.enago.jp) for their invaluable assistance with the English language review of this manuscript. Finally, we declare that all authors have no conflicts of interest.

REFERENCES

[1] F. D. Schneider, U. Brose, B. C. Rall, and C. Guill, “Animal diversity and ecosystem functioning in dynamic food webs,” *Nat. Commun. 2016 71*, vol. 7, no. 1, pp. 1–8, Oct. 2016, doi: 10.1038/ncomms12718.

[2] J. M. Marzluff, S. T. Knick, and J. J. Millspaugh, “High-Tech Behavioral Ecology: Modeling the Distribution of Animal Activities to Better Understand Wildlife Space Use and Resource Selection,” *Radio Track. Anim. Popul.*, pp. 309–326, Jan. 2001, doi: 10.1016/B978-012497781-5/50013-X.

[3] Sangam Babu Neupane, K. Sato, and B. P. Gautam, “A LITERATURE REVIEW OF COMPUTER VISION TECHNIQUES IN WILDLIFE MONITORING,” vol. 16, p. 2022, 2022, Accessed: Feb. 02, 2023. [Online]. Available: www.scientific-publications.net

[4] R. Kays *et al.*, "An empirical evaluation of camera trap study design: How many, how long and when?," *Methods Ecol. Evol.*, vol. 11, no. 6, pp. 700–713, Jun. 2020, doi: 10.1111/2041-210X.13370.

[5] M. W. Tobler, S. E. Carrillo-Percastegui, R. Leite Pitman, R. Mares, and G. Powell, "An evaluation of camera traps for inventorying large- and medium-sized terrestrial rainforest mammals," *Anim. Conserv.*, vol. 11, no. 3, pp. 169–178, Jun. 2008, doi: 10.1111/J.1469-1795.2008.00169.X.

[6] A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer, "Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna," *Sci. Data 2015 21*, vol. 2, no. 1, pp. 1–14, Jun. 2015, doi: 10.1038/sdata.2015.26.

[7] C. Affonso, A. L. D. Rossi, F. H. A. Vieira, and A. C. P. de L. F. de Carvalho, "Deep learning for biological image classification," *Expert Syst. Appl.*, vol. 85, pp. 114–122, Nov. 2017, doi: 10.1016/J.ESWA.2017.05.039.

[8] R. Ribani and M. Marengoni, "A Survey of Transfer Learning for Convolutional Neural Networks," *Proc. - 32nd Conf. Graph. Patterns Images Tutorials, SIBGRAPI-T 2019*, pp. 47–57, Oct. 2019, doi: 10.1109/SIBGRAPI-T.2019.00010.

[9] M. S. Norouzzadeh *et al.*, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 115, no. 25, pp. E5716–E5725, Jun. 2018, doi: 10.1073/PNAS.1719367115/-/DCSUPPLEMENTAL.

[10] S. B. Neupane and H. Youjirou, "Monitoring System to Minimize Wild Animal Hazards by Using Individual Organism Recognition Method with the Raspberry Pi and Computer Vision," 2022. Accessed: Apr. 20, 2022. [Online]. Available: https://www.researchgate.net/publication/360045868_Monitoring_System_to_Minimize_Wild_Animal_Hazards_by_Using_Individual_Organism_Recognition_Method_with_the_Raspberry_Pi_and_Computer_Vision

[11] A. Gomez Villa, A. Salazar, and F. Vargas, "Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks," *Ecol. Inform.*, vol. 41, pp. 24–32, Sep. 2017, doi: 10.1016/J.ECOINF.2017.07.004.

[12] S. Schneider, S. Greenberg, G. W. Taylor, and S. C. Kremer, "Three critical factors affecting automated image species recognition performance for camera traps," *Ecol. Evol.*, vol. 10, no. 7, pp. 3503–3517, Apr. 2020, doi: 10.1002/ECE3.6147.

[13] I. A. Zualkernan *et al.*, "Towards an IoT-based Deep Learning Architecture for Camera Trap Image Classification," *2020 IEEE Glob. Conf. Artif. Intell. Internet Things, GCAIoT 2020*, Dec. 2020, doi: 10.1109/GCAIOT51063.2020.9345858.

[14] Sara Beery, Grant Van Horn, and Pietro Perona, "Recognition in Terra Incognita," *Comput. Vis. - {ECCV} 2018 - 15th Eur. Conf.*, vol. 11220, 2018, doi: 10.1007/978-3-030-01270-0.

[15] T. Noble, C. Lortie, M. Westphal, H. B.- GigaScience, and undefined 2016, "A picture is worth a thousand data points: an imagery dataset of paired shrub-open microsites within the Carrizo Plain National Monument," *academic.oup.com*, 2016, doi: 10.1186/s13742-016-0145-2.

[16] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 20–29, 2004, doi: 10.1145/1007730.1007735.

[17] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0192-5.

[18] Z. Li, K. Kamnitsas, and B. Glocker, "Analyzing Overfitting under Class Imbalance in Neural Networks for Image Segmentation," *IEEE Trans. Med. Imaging*, vol. 40, no. 3, pp. 1065–1077, 2021, doi: 10.1109/TMI.2020.3046692.

[19] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.

[20] S. Imambi, K. B. Prakash, and G. R. Kanagachidambaresan, "PyTorch," *EAI/Springer Innov. Commun. Comput.*, pp. 87–104, 2021, doi: 10.1007/978-3-030-57077-4_10/COVER.

[21] N. Buduma, N. Buduma, and J. Papa, *Fundamentals of deep learning*. 2022. Accessed: Feb. 23, 2023. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=2e5vEAAAQBAJ&oi=fnd&pg=PT5&dq=fundamental+data+structure+in+deep+learning&ots=3YNawHeVDq&sig=175Md4REc9GM4bYHENr6zLKFle8

[22] Z. J. Wang *et al.*, "CNN 101: Interactive visual learning for convolutional neural networks," *Conf. Hum. Factors Comput. Syst. - Proc.*, Apr. 2020, doi: 10.1145/3334480.3382899.

[23] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arxiv.org*, Accessed: Feb. 23, 2023. [Online]. Available: https://arxiv.org/abs/2010.11929

[24] M. Tan, Q. L.-I. conference on machine learning, and undefined 2019, "Efficientnet: Rethinking model scaling for convolutional neural networks," *proceedings.mlr.press*, Accessed: Feb. 23, 2023. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

[25] F. Albardi, H. M. Di. Kabir, M. M. I. Bhuiyan, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A Comprehensive Study on Torchvision Pre-trained Models for Fine-grained Inter-species Classification," *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, pp. 2767–2774, Oct. 2021, doi: 10.1109/SMC52423.2021.9659161.

[26] E. Denton, A. Hanna, R. Amironesei, A. Smart, and H. Nicole, "On the genealogy of machine learning datasets: A critical history of ImageNet," *Big Data Soc.*, vol. 8, no. 2, 2021, doi: 10.1177/20539517211035955.

[27] V. Shankar, R. Roelofs, H. Mania, A. Fang, B. Recht, and L. Schmidt, "Evaluating Machine Accuracy on ImageNet," 2020.

[28] F. Luus, N. Khan, and I. Akhalwaya, "Active Learning with TensorBoard Projector," Jan. 2019, Accessed: Mar. 25, 2023. [Online]. Available: https://arxiv.org/abs/1901.00675v1

[29] T. Hoeser and C. Kuenzer, "Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends," *Remote Sens. 2020, Vol. 12, Page 1667*, vol. 12, no. 10, p. 1667, May 2020, doi: 10.3390/RS12101667.

[30] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, "A survey of transfer learning," *J Big Data.*, vol. 3, no. 1, p. 9, Dec. 2016, doi: 10.1186/s40537-016-0043-6.