

# Classification Taxonomies Genus of 90 Animals Using Transfer Learning Resnet-152

Satria Nur Saputro<sup>1</sup>, Faisal Dharma Adhinata<sup>2\*</sup>, and Ummi Athiyah<sup>3</sup>

<sup>1</sup>Department of Informatics, Faculty of Informatics, Institut Teknologi Telkom Purwokerto  
Jawa Tengah, Indonesia 53147

<sup>2</sup>Department of Software Engineering, Faculty of Informatics, Institut Teknologi Telkom Purwokerto  
Jawa Tengah, Indonesia 53147

<sup>3</sup>Department of Data Science, Faculty of Informatics, Institut Teknologi Telkom Purwokerto  
Jawa Tengah, Indonesia 53147

Email: <sup>1</sup>19102296@ittelkom-pwt.ac.id, <sup>2</sup>faisal@ittelkom-pwt.ac.id, <sup>3</sup>ummi@ittelkom-pwt.ac.id

**Abstract**—The process of learning theory and the limited ability to remember anything, especially a foreign language, often cause students to have difficulty understanding lessons, especially in determining the type and taxonomy of the animal. With the assistance of computer vision technology, students can more effectively face various challenges, enhance their understanding, and improve their ability to apply the concept of animal classification. The research classifies the taxonomy of 90 animals using Transfer Learning ResNet 152. It aims to analyze the performance of Transfer Learning ResNet 152 on the 90-animal dataset. The results show that in Model A with an architecture with frozen layers in 6 ResNet blocks, the highest evaluation value obtained is 0.9222 on Batch size 4 with Dropout 6, 0.9241 on Batch size 8 with Dropout 7, 0.9259 on Batch size 16 with Dropout 8, and 0.9296 on Batch size 32 with Dropout 4 and Dropout 7. Meanwhile, in model B with an architecture with frozen layers in 5 ResNet blocks and one non-frozen block, the highest evaluation value obtained is 0.7611 on Batch size 4 with Dropout 8, 0.8713 on Batch size 8 with Dropout 2, 0.8852 on Batch size 16 with Dropout 1, and 0.9204 on Batch size 32 with Dropout 3.

**Index Terms**—Taxonomy of Animals, Classification, Transfer Learning Resnet-152

## I. INTRODUCTION

**I**MAGE classification is a machine learning technique that quickly and accurately recognizes, identifies, and distinguishes images. It is closely related to recent advancements in image classification research, such as data augmentation and optimization methods, which can significantly improve various Convolutional Neural Network (CNN) models, particularly in transfer learning performance [1].

Received: Jan. 19, 2023; received in revised form: Aug. 02, 2023; accepted: Aug. 03, 2023; available online: April 05, 2024.

\*Corresponding Author

Animal taxonomy is a field in biology that studies how to group, classify, and systematically arrange animals based on their similarities and differences in characteristics. Its goal is to provide names to species, classify existing species in nature, and understand their evolutionary relationships [2]. In biology, students typically have to memorize the taxonomy sub-chapter about the classification of vertebrate animals. However, some students struggle to remember information and understand the lessons in theoretical and foreign language courses, especially identifying animal species [3]. Taxonomy is also defined through searches from various sources on the Internet, starting from kingdom, phylum, class, order, family, and genus.

Deep learning is a branch of machine learning that uses artificial neural networks inspired by the human brain's cortex with Artificial Neural Network (ANN) algorithms and more complex hidden layers [4]. CNN is one of the deep learning methods used for highly accurate object image classification, combining intra-network architecture and inter-architecture network fusion [4–7]. However, it requires a long training time, so Graphics Processing Unit (GPU) is recommended to accelerate computational performance [8].

Previous research has used ResNet 50 and 152 for pneumonia detection in humans with a dataset of 3,468 pneumonia and normal images. It has the best results at epoch 150 with an accuracy of 88.7% and 89.3% on ResNet 50 and 152, respectively [9]. Another research has classified faces into happy, sad, fearful, angry, surprised, disgusted, and neutral using a dataset of 700 images with a size of 100×100 pixels. They have modified the ResNet 152 model to retrain the dataset's weights, including batch normalization, Microsoft Research Asia + Batch Normaliza-

tion (MSRA+BN) initialization, and others. The model has achieved impressive classification success with an accuracy rate of 96.44% [10].

From a different source, gender classification through retinal images has been conducted using a dataset from the Vancouver General Hospital Department of Ophthalmology with the initials DOVS-i, DOVS-ii, ODIN-N, and ODIN-C. It has modified the ResNet 152 architecture, achieving an Area Under the Curve (AUC) score of 0.72 (95%) using approximately 2,500 images [11, 12]. In another research, the classification of four Genera Panthera using CNN, a dataset of 5,600 images, divided into 3,840 for training, 960 for validation, and 800 for testing, has been resized to  $128 \times 128$  and normalized to 0-1, with CNN architecture and 100 epochs, a batch size of 64, and a training time of 46 minutes. It has obtained a training accuracy of 92.31% and a validation accuracy of 81.88% [13].

Next, transfer learning is a machine learning method that uses pre-determined network features to solve different problems. It is efficient in training, saves computational costs in training new networks, and improves generalization in solving new situations [14]. Residual Network (ResNet), developed by Microsoft, is a transfer learning technique that solves the problem of vanishing gradients in the ImageNet 2015 contest by using skip connections on some convolutional neural network layers [15].

The research uses ResNet 152 to classify animal images to solve animal taxonomy problems. ResNet performs well in convolutional networks with a top-five error rate of only 3.57%, better than other ResNet units. Despite having fewer parameters than Visual Geometry Group Network (VGGNet), the concept of residual learning introduced by ResNet is also effective in addressing network degradation issues. This architecture is very good for optimization and high network depth accuracy [9, 16]. It is hoped that the research results will provide information about animal taxonomy that can be used as a reference for further analysis using transfer learning ResNet 152 with the multiclass dataset to solve the problem of animal genus taxonomy, especially with animal image data.

## II. RESEARCH METHOD

### A. Data Collection

The dataset is from the Kaggle (<https://www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals>). It has 5,400 images and 90 types of animals. Each class consisted of 60 Joint Photographic Experts Group (JPG) format images with various image dimensions.

### B. Data Preprocessing

Data preprocessing is a technique used to collect raw data in a format and color format and resize images to be efficient for training. In the research, the images are resized to the dimensions of 244 by 244 pixels. It standardizes the image size and likely helps in reducing computational load. Additionally, the color format of the images is changed to Blue, Green, and Red (BGR), which is a common color space used in various image processing applications.

### C. Splitting Data for Training and Testing

The dataset is divided into training and test sets for training and validation, with a ratio of 80% for training and 20% for testing. It means that 4,320 images are used for training data, and 1,080 images are used for validation data. This split is a standard practice in deep learning to ensure that the model can be trained on one portion of the data (the training set) and evaluated on a separate portion (the testing set) to validate its predictive performance.

### D. Data Augmentation

Data augmentation is a technique aiming to enhance the diversity of a training dataset by applying realistic and random transformations to the images. This process is crucial for improving the robustness and accuracy of machine learning models, particularly in scenarios where acquiring additional data is impractical or cost prohibitive. The essence of data augmentation lies in its ability to modify an image in such a way that it is perceived as a different image by the model without altering its fundamental content or category. For instance, by applying random tilts or rotating the image horizontally by up to 10%, an image of a cat can be transformed in appearance while still maintaining its identity as a cat. These transformations introduce variations in orientation, simulating the diversity of perspectives that a model may encounter in real-world scenarios. By training on this augmented dataset, models can learn to recognize objects or features regardless of their orientation, significantly reducing the risk of overfitting. This practice ensures that machine learning models are not only more versatile but also more accurate when presented with new, unseen data.

### E. Transfer Learning

The research uses the Pre-Trained ResNet 152 architecture for classifying new objects through transfer learning and data augmentation to avoid excessive layers and underfitting. The transfer learning process goes through the convolutional, max pooling, and

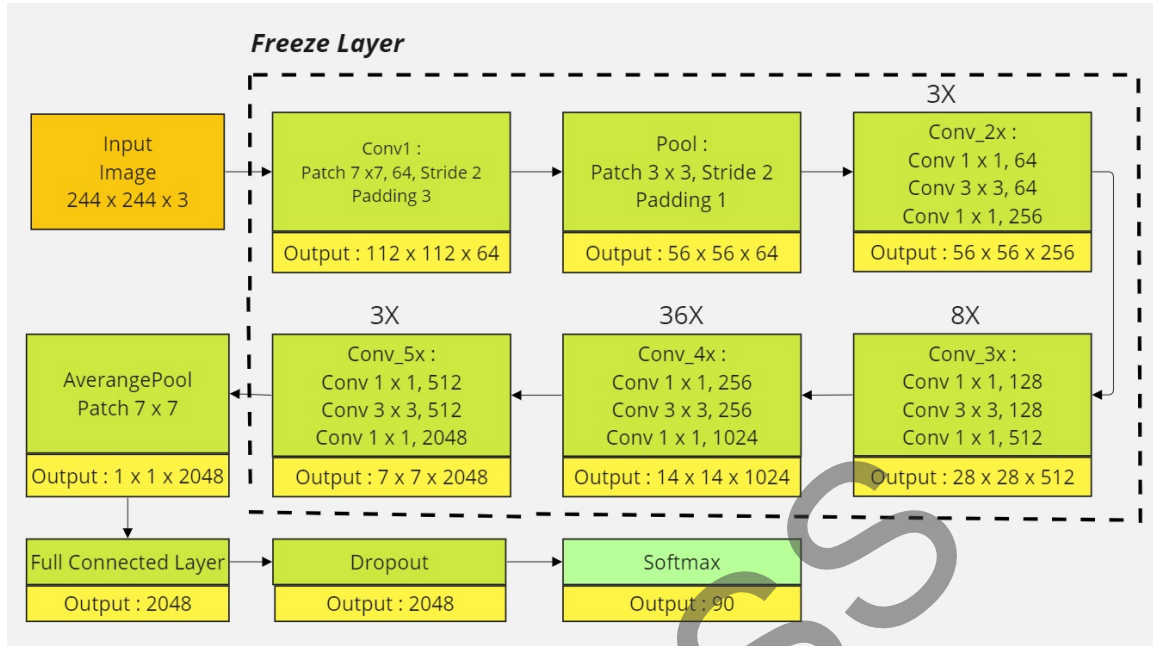


Fig. 1. Model A.

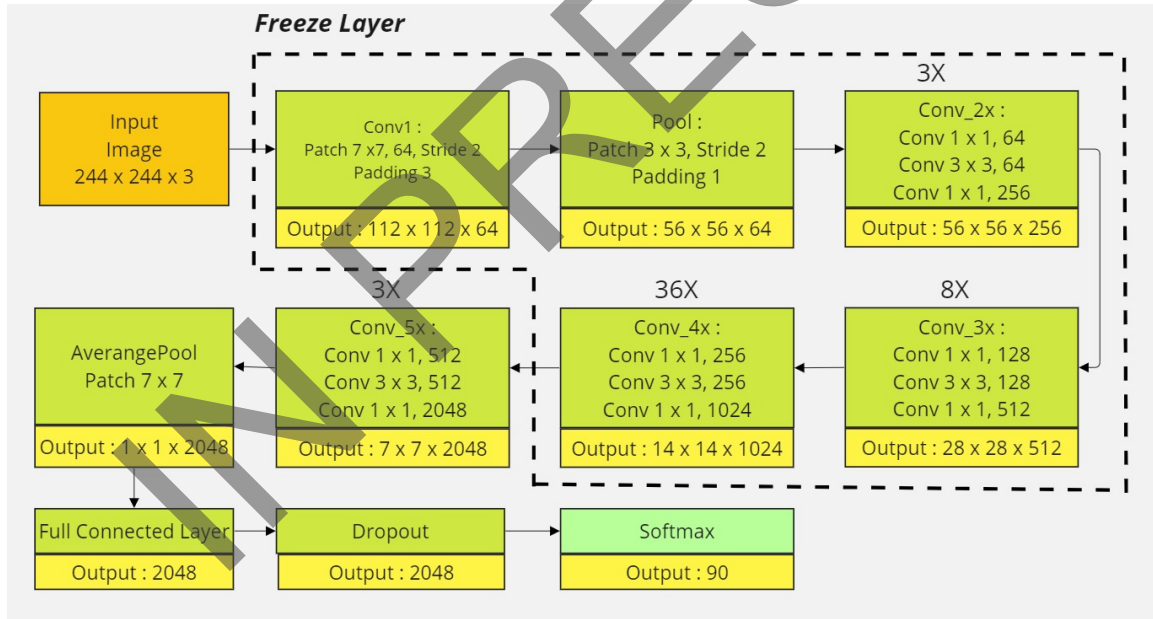


Fig. 2. Model B.

residual block processes to produce a 90-dense layer with a Softmax activation layer. The research uses two models.

Figure 1 illustrates the architecture scheme of Model A. It utilizes freeze layers entirely, starting from Conv1, Conv\_2x, Conv\_3x, Conv\_4x, and Conv\_5x, for the training architecture as the implementation of Transfer Learning with ResNet 152. Freeze layers aim

to preserve the learned feature representations from the pre-trained Model for use with new objects by keeping the weight values fixed.

Figure 2 illustrates the architecture of model B. It uses freeze layers entirely, starting from Conv1, Conv\_2x, Conv\_3x, and Conv\_4x, except for the last 3x block (Conv\_5x), which is in trainable mode. Then, Conv\_5x updates its weights, allowing the model to

adapt the feature representations to new data in the dataset.

#### F. Formulas Used

- 1) ResNet [15] is formulated by Eq. (1). It has  $H(x)$  as the expected weight layer, resulting from  $F(x) + x$ . Then,  $F(x)$  is the weight layer, and  $x$  is the identity

$$H(x) = F(x) + x. \quad (1)$$

- 2) The pooling layer reduces the input matrix size by decreasing the number of parameters, with three types: min pooling, max pooling, and average pooling [14].
- 3) The convolution layer is an input layer for activating feature extraction on an image with convolution filter calculations, with Eq. (2). It consists of  $h(x)$  as the weight parameter of the convolution calculation of  $f(x)$  as the image input and  $g(x)$  as the filter.

$$h(x) = f(x) * g(x). \quad (2)$$

- 4) Batch normalization is a method of the output layer that speeds up the learning process by preventing extreme activation values and functions as a regularize or dropout [17] and 4-dimensional tensor layer, where  $b$  is the batch,  $c$  is the channel, and the two spatial dimensions are  $x$  and  $y$  [18], with the Eq. (3). Then, the average activation derivative of batch normalization is formulated by Eq. (4). From all batch normalization inputs, it divides the centered activation by the standard deviation. During testing, mean and variance are used, and normalization is transformed into parameters  $\gamma_c, \beta_c$  during training.

$$o_{b,c,x,y} = \gamma \frac{I_{b,c,x,y} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c, \quad (3)$$

$$\mu_c = \frac{1}{|\beta|} \sum_{b,x,y} I_{b,x,y}. \quad (4)$$

- 5) Rectified Linear Unit (ReLU) is an activation layer that maps negative values to 0 and maintains positive values, formulated by Eq. (5). At the lowest value of 0, the activation layer will output 0 if it is  $x < 0$  or  $x$  if it is  $x > 0$ .

$$f(x) = \max(0, x). \quad (5)$$

- 6) The Fully Connected layer is a layer that connects all the layers called feature maps and inputs them into the Fully Connected layer [19]. It is

formulated by Eq. (6). It has  $g$  as the activation layer,  $b$  as the bias,  $w_i$  as the input value, and  $x_i$  as the weight value.

$$h(x) = g \left( b + \sum_i w_i x_i \right). \quad (6)$$

- 7) The Dropout layer is a regularization method that deactivates neurons randomly to avoid overfitting during training by randomly selecting neurons and preventing using weights during backpropagation [20]. Equation (7) does the feedforward process. It shows  $y^l$  as the value of output layer  $l$ ,  $z^l$  as the value of input layer  $l$ , and  $W^l$  and  $b^l$  as the weights of layer  $l$ , using unit  $i$  with activation function  $f$  and  $r^l$  vector until  $j$  stores the values obtained from the Bernoulli distribution.

$$\begin{aligned} y^{l+1} &= r_j^l * y^l, \\ z_i^{l+1} &= W_i^{(l+1)} y^l + b_i^{(l+1)}, \\ y_i^{l+1} &= f \left( z_i^{(l+1)} \right). \end{aligned} \quad (7)$$

- 8) The Softmax activation layer converts values to probability classes using the Softmax function to obtain classification results [21]. It is formulated by Eq. (8). It includes  $y_{ijk}$  as a vector of values between 0–1 and  $x$  as a vector containing the last Fully Connected layer.

$$y_{ijk} = \frac{e^{x_{ijk}}}{\sum_{t=1}^D e^{x_{ijt}}}. \quad (8)$$

- 9) Adam optimizer, a combination of Root Mean Square Propagation (RMSprop) and momentum, results in the derivative of the Stochastic Gradient Descent (SGD) method initially created from the adaptive estimation of orders 1 and 2. Thus, the learning is adaptive for each parameter and stores the average gradient value. The learning rate is  $3e^{-4}$ , and the Adam optimizer calculation formula is in Eq. (9). It consists of  $\theta_{(t+1)}$  as the result of the new parameter,  $\theta_t$  as the parameter before being updated,  $\partial$  is the learning rate,  $\hat{m}_t$  is the order one squared gradient,  $\hat{v}_t$  as the order of two gradients, and  $\epsilon$  as a small-scale prevention of division by 0. The error calculation directly depends on the order 1 and 2 gradients.

$$\theta_{t+1} = \theta_t - \frac{\partial}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t. \quad (9)$$

- 10) The loss measures the models' error during training and testing by using categorical cross entropy as the optimization value of the loss equation (see Eq. (10)).  $H$  is the loss value,  $p$  is the original value,  $q$  is the estimated value,  $p_i$  is the value

of the original probability (class), and  $q_i$  is the estimated probability value (Softmax) [22].

$$H(p, q) = -\sum_{i=1}^N p_i \log(q_i). \quad (10)$$

- 11) The accuracy metric is used to measure the models' performance from the testing data and the loss and accuracy value in every training iteration to determine underfitting [23].

#### G. Model Evaluation

Equation (11) tests accuracy. The  $A$  is the value of the percentage of accuracy. Meanwhile,  $b$  is the number of correct data, and  $n$  is the total amount of data.

$$A = \frac{b}{n} \times 100\%. \quad (11)$$

#### H. Visualization

The research utilizes the Pandas and Matplotlib libraries to create visualizations in the form of images for the accuracy results, validation accuracy, loss, and validation loss. Leveraging the capabilities of the Pandas and Matplotlib libraries is essential to analyze and interpret the performance of deep learning models effectively. Data related to accuracy results, validation accuracy, loss, and validation loss can be efficiently organized and manipulated for analysis using Pandas. Meanwhile, Matplotlib provides a robust platform for creating various visualizations, from simple plots to complex images, enabling a clear graphical representation of the model's performance metrics. These visualizations facilitate a deeper understanding of the model's behavior over various epochs, highlighting trends and potential areas for improvement in both the training and validation phases.

#### I. Flow Diagram

The research follows a flow diagram, a series of steps to achieve the research goals, as shown in Fig. 3. The process commences with the literature review, which entails reading and summarizing the existing knowledge and findings pertinent to the topic. It is succeeded by formulating the problem, a stage in which the specific issue or challenge the research aims to address is clearly defined. The subsequent phase is data collection, a pivotal stage where relevant data is amassed to tackle the identified problem. This step includes a comprehensive breakdown of tasks such as system requirements analysis, data preprocessing, splitting data for training and testing, data augmentation, transfer learning, model creation, model evaluation,

and visualization. It culminates with Python analysis, which implies the use of the Python programming language for analytical purposes. After collecting data, the workflow advances to system implementation and testing, during which the proposed solution or model is constructed and subjected to rigorous testing to verify that it fulfills the stipulated requirements and effectively resolves the problem. Prior to concluding the process, the final phase is hypotheses, conclusions, and suggestions. In this crucial stage, hypotheses are either substantiated or refuted, inferences regarding the research findings or project results are articulated, and recommendations for prospective endeavors, enhancements, or associated studies are proposed.

#### J. Architecture Model

Figure 4 illustrates the deep learning architecture for Model A, in which all layers are frozen. For Model B, all layers are frozen except for the last three blocks used for testing in the research. It begins with an "Input Layer" that accepts images of size  $224 \times 224$  pixels with three color channels (representing RGB). The data are then passed through a ResNet152 layer, a pre-trained neural network that processes the image and outputs a much-reduced spatial dimension of  $7 \times 7$ , but with 2,048 filters, capturing a wide array of features. Following this process, an "AveragePooling2D" layer further condenses the data by performing average pooling, resulting in a single  $1 \times 1$  spatial dimension with 2,048 filters. The Flatten layer then converts this multi-dimensional output into a one-dimensional array of size 2,048. A Dropout layer follows, which helps prevent overfitting by randomly setting a fraction of the input units to zero during training. Finally, the Dense layer, with a 'Softmax' activation function, takes the output from the Dropout layer and produces a probability distribution across 90 classes, indicating the models' predictions for the input image.

#### K. Research Scenario

The research scenario is shown in Fig. 5. The experiment starts with a dataset that is divided into different batch sizes: 4, 8, 16, and 32. Model A and Model B are trained separately for each batch size. After training, a dropout technique is applied to each model. The notation "Dropout 1-8" suggests that a range of dropout rates from 1 to 8 is tested. Dropout is a regularization technique used to prevent overfitting in neural networks by randomly omitting a proportion of features during training. By systematically varying the batch sizes and applying dropout rates, the experiment aims to observe how these factors affect the learning and generalization capability of the models. This setup

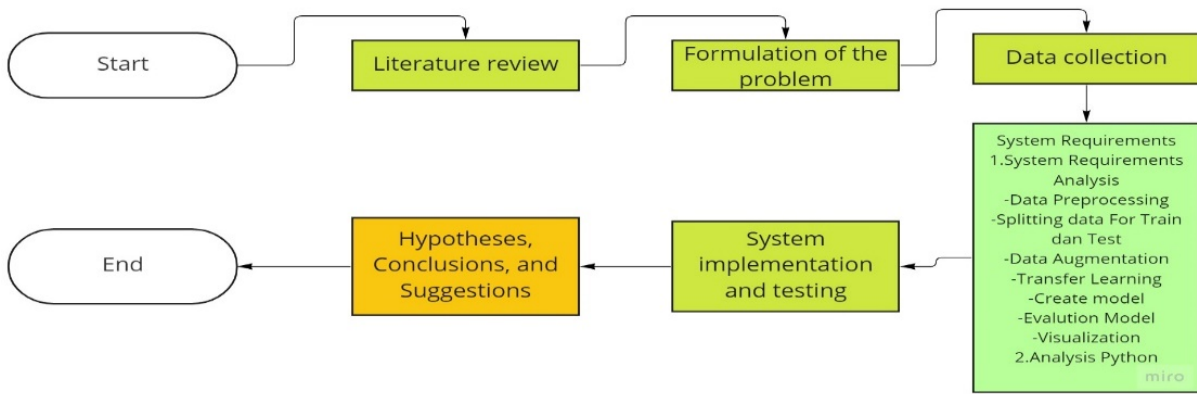


Fig. 3. Flow diagram.

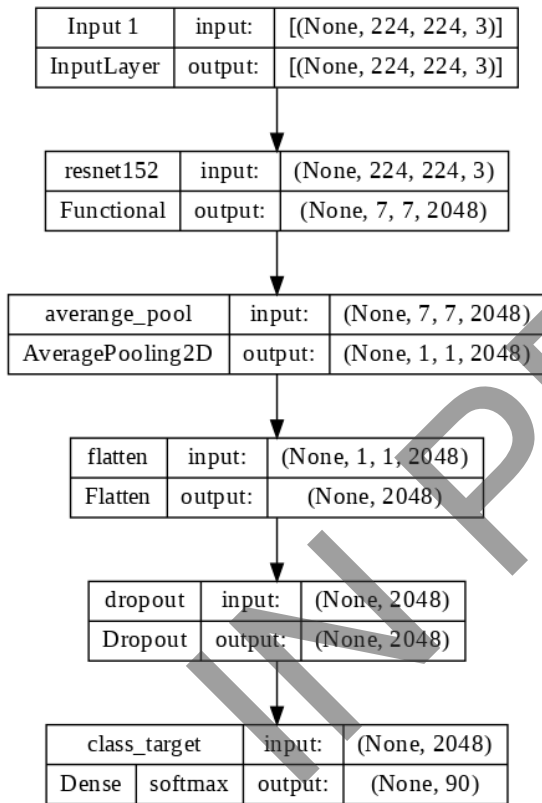


Fig. 4. Deep learning architecture.

allows for a comprehensive analysis of the model's robustness and accuracy across different training conditions, helping to identify optimal configurations for these models when processing the given dataset.

#### L. Architecture of Model A

The research uses the architecture of Model A, as shown in Fig. 6. The summary outlines the architecture

and parameter details of the model, beginning with a ResNet152 layer. This pre-trained convolutional neural network functions as the foundational feature extractor. This layer outputs a tensor with dimensions (None, 7, 7, 2048) and contributes a substantial 58,370,944 non-trainable parameters to the model. Following it is an 'Average Pool' layer, which applies average pooling to condense the feature maps into a shape of (None, 1, 1, 2048), adding no new parameters as pooling operations do not involve learning. The subsequent 'Flatten' layer transforms the pooled feature maps into a one-dimensional array of 2,048 elements, a necessary step before the data enters the final layers. Next, a 'Dropout' layer is included to mitigate overfitting by randomly omitting a portion of the input units during the training phase, and it also does not have trainable parameters. The last layer, named 'Class Target', is a densely connected layer with 90 neurons that corresponds to the number of target classes for the model, equipped with a Softmax activation to yield a probability distribution for class predictions. This layer introduces 184,410 trainable parameters.

#### M. Architecture of Model B

The research also uses the architecture of Model B, as shown in Fig. 7. The first layer, 'ResNet152', is a functional layer known for its depth and complexity, designed to extract features from input images. This layer outputs a tensor with a shape of (None, 7, 7, 2048) and has 58,370,944 parameters. All of them are non-trainable, implying that they are pre-set and not subject to change during training. Following it is the 'Average Pool' layer, an AveragePooling2D operation that simplifies the output by reducing its dimensions to (None, 1, 1, 2048). This pooling layer has no trainable parameters. Next, the 'Flatten' layer converts the pooled 2D feature maps into a 1D tensor, specifically

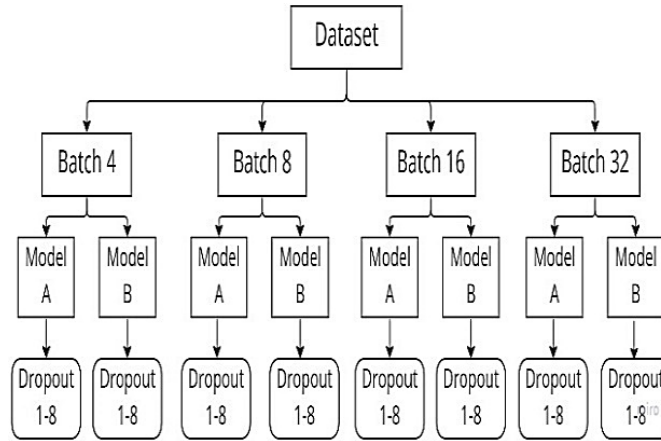


Fig. 5. Research scenario.

Layer (type)	Output Shape	Param #
resnet152 (Functional)	(None, 7, 7, 2048)	58370944
average_pool (AveragePooling2D)	(None, 1, 1, 2048)	0
flatten (Flatten)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
class_target (Dense)	(None, 90)	184410
=====		
Total params: 58,555,354		
Trainable params: 184,410		
Non-trainable params: 58,370,944		
=====		
CPU times: user 7.23 s, sys: 1.23 s, total: 8.46 s		
Wall time: 15.6 s		

Fig. 6. Architecture of Model A.

Layer (type)	Output Shape	Param #
resnet152 (Functional)	(None, 7, 7, 2048)	58370944
average_pool (AveragePooling2D)	(None, 1, 1, 2048)	0
flatten (Flatten)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
class_target (Dense)	(None, 90)	184410
=====		
Total params: 58,555,354		
Trainable params: 15,160,410		
Non-trainable params: 43,394,944		
=====		
CPU times: user 7.04 s, sys: 1.33 s, total: 8.37 s		
Wall time: 20.2 s		

Fig. 7. Architecture of Model B.

a vector with 2,048 elements, preparing the features for the subsequent layers without adding additional parameters. The 'Dropout' layer is then employed to help prevent overfitting. It randomly sets a fraction of

the input units to zero during training, and consistent with the nature of dropout operations, it does not have any trainable parameters. The final layer, 'Class Target', is a densely connected (Dense) layer with a



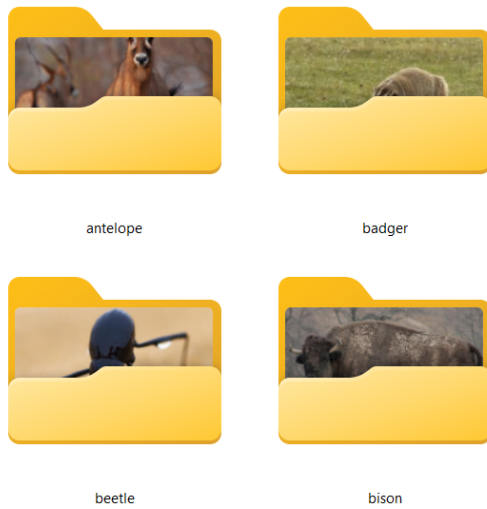


Fig. 8. Example of datasets.

Softmax activation function, outputting a probability distribution over 90 possible classes. This layer has 184,410 trainable parameters.

### III. RESULTS AND DISCUSSION

The research uses a dataset consisting of images from the Kaggle website. Figure 8 is an example of a dataset folder, which serves as an example for managing and accessing data related to animal taxonomy. A comprehensive search to compile this dataset is conducted across various sources on the Internet to gather information on a diverse array of animal species. The scope of the search is focused on identifying 90 distinct animal species. These species are selected based on predefined categories within the dataset. This methodical approach to data collection ensures that the dataset is rich and varied, encompassing a wide spectrum of animal life.

Then, the research performs dataset splitting as described in Fig. 9. The dataset is initially partitioned into two segments, with 80% allocated for training and the remaining 20% for validation. Each subset contains 90 folders, which suggests 90 distinct categories or classes within the dataset. This structure is typical in deep learning, where a model is trained on a large portion of the dataset (in this case, 80%) and evaluated on a smaller and separate portion (here, 20%) to assess its performance.

Table I serves as an illustrative reference for understanding the taxonomic distinctions between different animal species. The animals listed are the Antelope, Badger, Bat, Bear, and Bee. All animals belong to the Kingdom Animalia, with the first four also sharing

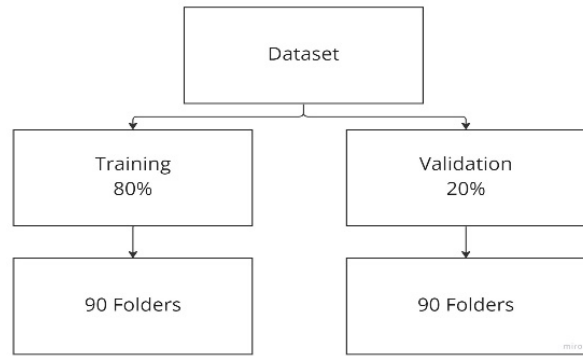


Fig. 9. Splitting datasets.

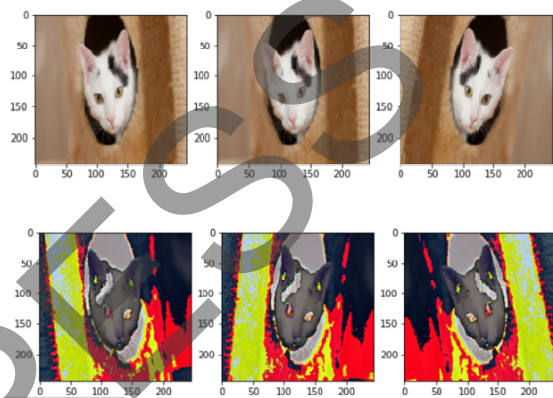


Fig. 10. Resizing 244×244 format of Red, Green, Blue (RGB) to Blue, Green, and Red (BGR).

the Phylum Chordata and Class Mammalia, differing in their Order, Family, and Genus classifications. The Bee differs more substantially, belonging to the Phylum Arthropoda and Class Insecta, reflecting its distinct evolutionary lineage compared to the mammals listed.

The next step is data preprocessing. Data preprocessing is a crucial step that often precedes the actual analysis or modeling. In this particular context, preprocessing involves standardizing the size of the images to ensure uniformity, which is essential for the algorithm's consistent processing and analysis by the algorithm. The specified dimensions to which the images are resized are 244 by 244 pixels, as shown in Fig. 10. Moreover, the color format of the images is adjusted to BGR.




The subsequent columns of Table II demonstrate the augmentation techniques. Horizontal flip displays the image after it has been flipped along the vertical axis, creating a mirror image. Rotation 10° shows the same original image rotated slightly by 10 degrees. These visual modifications serve to artificially expand the dataset, which can improve the robustness of a neural



TABLE I  
EXAMPLE OF FIVE ANIMAL TAXONOMIES.

No	Name	Taxonomy					
		Kingdom	Phylum	Class	Order	Family	Genus
1	Antelope	Animalia	Chordata	Mammalia	Artiodactyla	Mustelidae	Antelope
2	Badger	Animalia	Chordata	Mammalia	Carnivora	Bovidae	Mellivora
3	Bat	Animalia	Chordata	Mammalia	Chiroptera	Pteropodidae	Pteropus
4	Bear	Animalia	Chordata	Mammalia	Carnivora	Ursidae	Ursus
5	Bee	Animalia	Arthropoda	Insecta	Hymenoptera	Apidae	Apis

TABLE II  
EXAMPLE OF DATA AUGMENTATION (RED, GREEN, BLUE (RGB)).

Name of Data	Raw Data	Horizontal Flip	Rotation 10%
Cat			

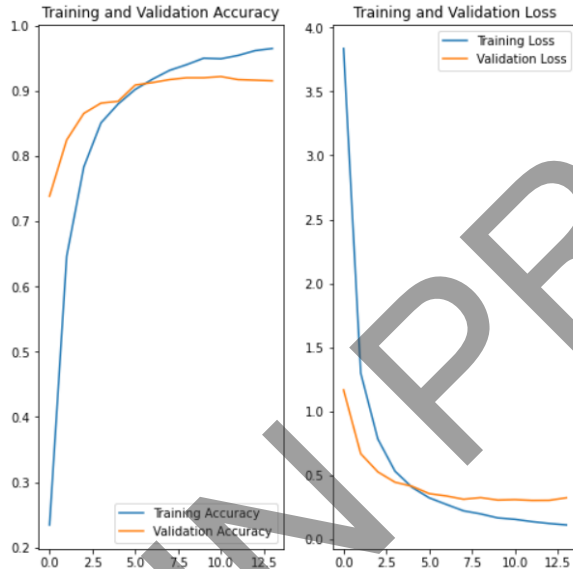


Fig. 11. The best visualization in Model A with batch size 4 and dropout 6.

network by training it on varied versions of the same image. Hence, it can help the model to generalize better when encountering new data. The data augmentation process is crucial in preventing overfitting and enhancing the models' ability to recognize images in different orientations and configurations.

After the previous steps, the ResNet 152 transfer learning algorithm uses the dataset to produce a model that can classify the animal type and genus taxonomy. In the research, two models are used: Model A with all layers frozen, referring to Fig. 1, and Model B with all layers frozen except for the last three blocks, referring

to Fig. 2. The optimizer used is Adam. The loss function used is also Adam, the loss function is categorical cross entropy, and metrics are accuracy. Moreover, the callback is saved as the best model based on validation accuracy mode maximum, Comma Separated Values Logger (CSV Logger) to record per-iteration results, and early stopping validation loss mode minimum with the patience of two restore weights.

As seen in Fig. 11, at epoch 14, the training accuracy reaches 0.9653, indicating a high level of proficiency on the training dataset. Meanwhile, the validation accuracy achieved is slightly lower at 0.9157, suggesting that the model also performs well on the unseen validation dataset. However, it is not quite as well as on the training set, which is a common occurrence in machine learning due to overfitting the training data. Regarding the loss metrics, which are indicative of the error or the cost function used during training, the training loss is relatively low at 0.1117 by epoch 14, showing that the model is a good fit for the training data. The validation loss is higher at 0.3241, which is expected as the validation data presents new challenges to the model.

Figure 12 produces epoch 19. At this stage, the training accuracy reaches 0.9389, suggesting that the model correctly predicts the training data with high reliability. The validation accuracy, at 0.9241, is slightly lower but still relatively high, indicating that the model generalizes well to new or unseen data. On the loss graph, the training loss—representing the model's error on the training dataset—is reported as 0.1937, which is relatively low, showing that the model's predictions are fairly close to the actual values. The validation loss is higher at 0.3057, which is common as the validation

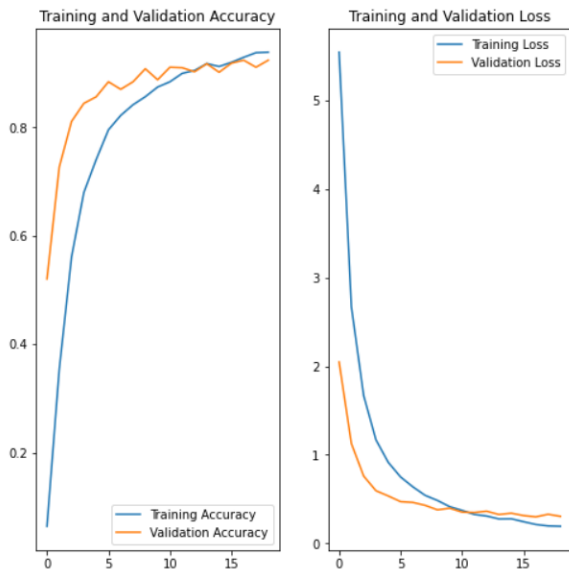


Fig. 12. The best visualization of Model A with batch size 8 and dropout 7.



Fig. 14. The best visualization of Model A with batch size 32 and dropout 7.

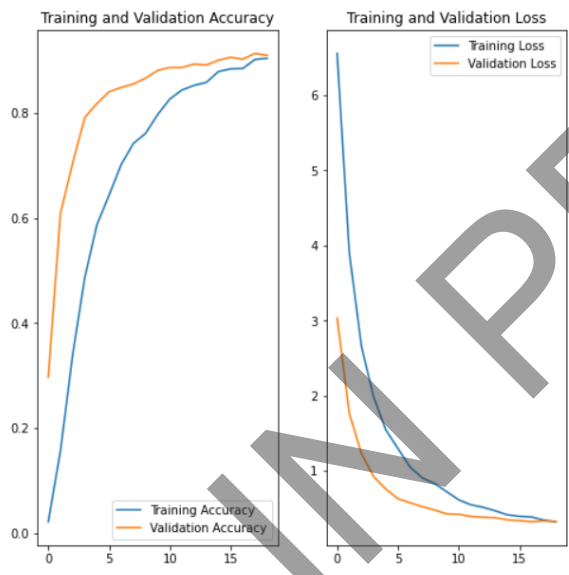


Fig. 13. The best visualization of Model A with batch size 16 and dropout 8.

set is not used during the training phase, making it a good indicator of how the model might perform on data outside the training dataset.

According to Fig. 13, by epoch 24, which may be the last epoch in the training session, the model achieves a training accuracy of 0.8968. This percentage reflects the proportion of the training dataset that the model can correctly classify. Concurrently, the validation accuracy—a measure of the model’s performance on a separate dataset not seen during training—stands

at a slightly higher 0.9259. This result suggests that the model generalizes well to new data, which is a positive indicator of its predictive capabilities. In terms of loss, which quantifies the error between the model’s predictions and the actual values, the training loss at epoch 24 is 0.3255. This value indicates the average degree to which the model’s predictions deviate from the target values in the training data. The validation loss records a similar figure of 0.2891, which is marginally lower than the training loss, implying that the model’s predictions are slightly closer to the actual values in the validation set than in the training set at epoch 24.

Figure 14 at epoch 38 shows the model’s training accuracy—the percentage of the training dataset correctly classified by the model—reported as 0.9315. This result reflects a high degree of accuracy and suggests that the model has learned to effectively recognize patterns in the training data. In parallel, the validation accuracy, which measures how well the model performs on a separate dataset not seen during training, is slightly lower at 0.9269. This close proximity of training and validation accuracy is a positive indicator because it suggests that the model generalizes well and does not overfit the training data. On the loss side, the training loss at epoch 38 is noted as 0.2215. This figure represents the average error between the model’s predictions and the actual target values for the training dataset, and a lower number is preferable. The validation loss, at 0.2815, is slightly higher than the training loss but still relatively low, which again indicates that the model

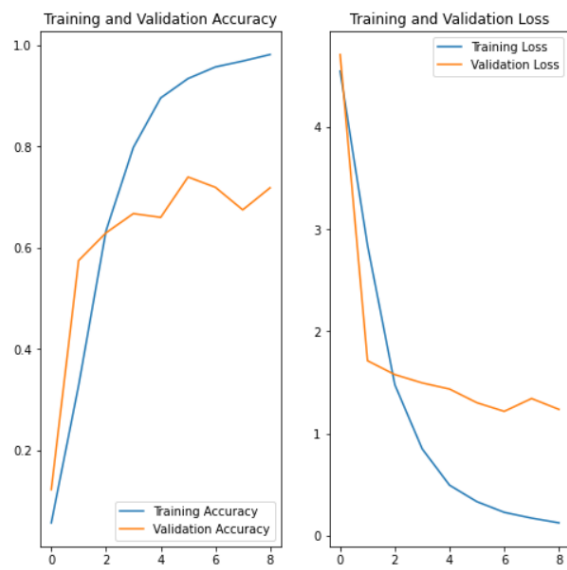


Fig. 15. The best visualization of Model B with batch size 4 and dropout 8.

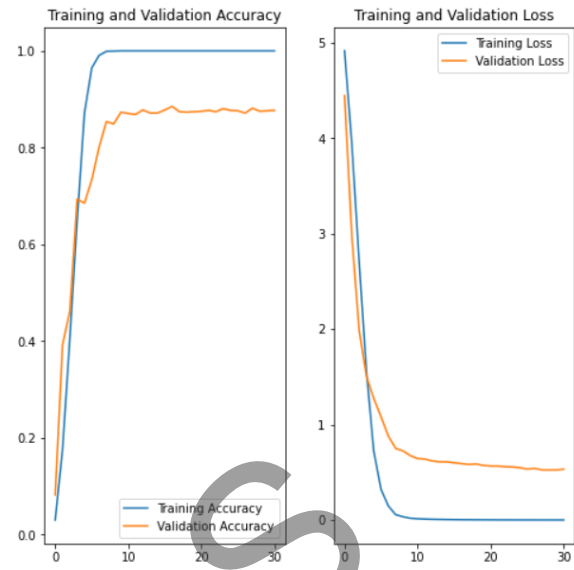


Fig. 17. The best visualization of Model B with batch size 16 and dropout 1.

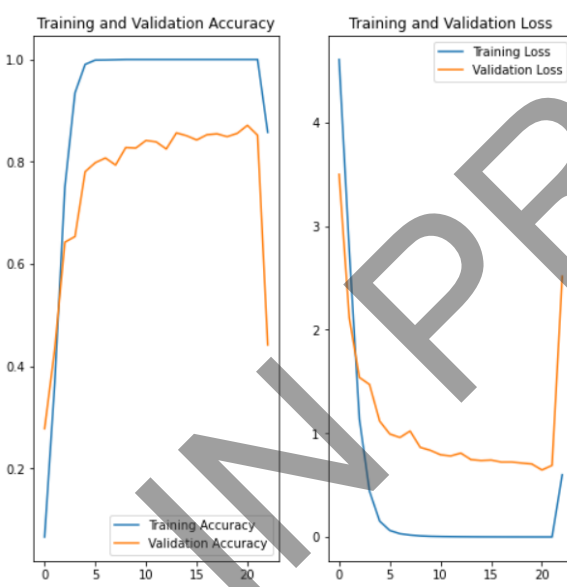


Fig. 16. The best visualization of Model B with batch size 8 and dropout 2.

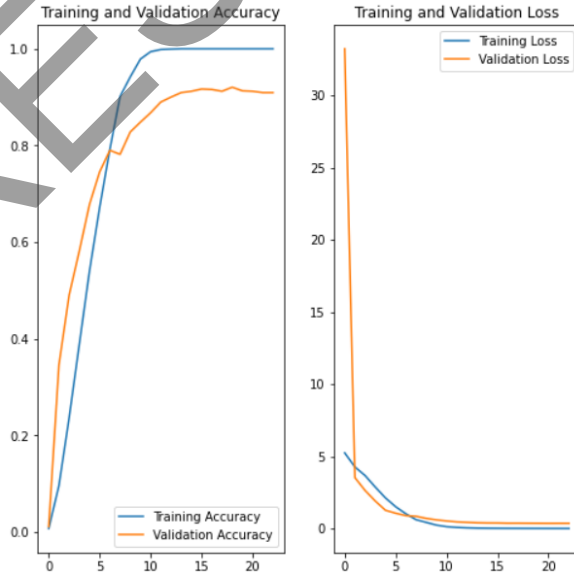


Fig. 18. The best visualization of Model B with batch size 32 and dropout 3.

generalizes well to unseen data.

According to Fig. 15, at epoch 7, the model reaches a training accuracy of 0.866, which means that it is able to correctly predict the outcome for a significant majority of the training set. However, the validation accuracy is lower at 0.7306, indicating a less successful rate of correct predictions on the validation set. It consists of data not seen by the model during training. This discrepancy can often signal overfitting, where a model

performs well on the data it has seen (training data) but poorly on new and unseen data (validation data). In terms of loss, which quantifies the difference between the model's predictions and the actual outcomes, the training loss stands at 0.5441 at epoch 7. This value is moderately low, but when compared to the validation loss, which is considerably higher at 1.288, it suggests that the model's predictions are less accurate on the validation set than on the training set.

Figure 16, at epoch 23, shows that the training

accuracy of the model is approximately 0.8576, indicating a high success rate in correctly predicting the outcomes for the training dataset. Conversely, the validation accuracy is significantly lower at 0.4417, suggesting that the model does not perform nearly as well on the validation set, which is a separate data subset not seen during training. Such a stark difference between training and validation accuracy often points to overfitting, where a model is tuned too closely to the training data and fails to generalize effectively to new data. The training loss at epoch 23 is reported to be 0.6005 looking at the loss metrics. This loss value indicates the average error magnitude in the model’s predictions compared to the actual target values in the training set. In contrast, the validation loss is much higher at 2.5158, revealing that the model’s predictions are significantly less accurate on the validation set.

In Fig. 17 at epoch 31, the model achieves a perfect training accuracy of 1.0, indicating that it can correctly classify every instance in the training dataset. However, the validation accuracy is lower, at approximately 0.8769, which suggests that while the model performs exceptionally well on the training data, it is less accurate when predicting outcomes on the validation dataset. This discrepancy often signals overfitting because it implies that the model may have memorized the training data rather than learned generalizable patterns applicable to unseen data. In terms of the loss values, the training loss is extraordinarily low at 0.0003, which indicates that the model’s predictions are almost exactly in line with the actual targets in the training set. The validation loss, on the other hand, is significantly higher at 0.5320. While not excessively high in absolute terms, when compared to the training loss, it suggests that the model’s predictions for the validation set are less precise.

Figure 18 at epoch 23, the model achieves a training accuracy of 1.0, which means it has learned to classify the training data with no errors. However, the validation accuracy stands at 0.9093, which is lower than the training accuracy but still quite high. This result shows that the model can generalize well to new data, although not perfectly. Perfect training accuracy can also indicate potential overfitting, especially if the validation accuracy has been significantly lower. The training loss, which measures how well the model’s predictions match the actual labels during training, is extremely low at 0.0058, supporting the high training accuracy. The validation loss is higher at 0.3628 but is still within a reasonable range, suggesting the model’s predictions are fairly close to the actual values in the validation set as well.

Table III provides a detailed performance evaluation of Model A when trained with a batch size of 4,

TABLE III  
MODEL A WITH BATCH SIZE 4.

No	BATCH 4				
	Accuracy	Loss	Evaluation	Epochs	Time (s)
	Last Epochs				
D1	0.9968	0.0222	0.9185	11	2847
D2	0.9854	0.0646	0.9185	14	3424
D3	0.9914	0.0398	0.9111	12	2396
D4	0.9801	0.0835	0.9120	9	3439
D5	0.9789	0.0776	0.9204	12	4008
D6	0.9653	0.1117	0.9222	14	5614
D7	0.9317	0.2221	0.9194	14	7216
D8	0.8303	0.5899	0.9065	10	3470

TABLE IV  
MODEL A WITH BATCH SIZE 8.

No	BATCH 8				
	Accuracy	Loss	Evaluation	Epochs	Time (s)
	Last Epochs				
D1	0.9968	0.0252	0.9213	17	4157
D2	0.9907	0.0584	0.9148	13	5062
D3	0.9884	0.0571	0.9231	15	4184
D4	0.9875	0.0572	0.9194	18	5713
D5	0.9690	0.1174	0.9157	14	3124
D6	0.9558	0.1546	0.9185	15	2794
D7	0.9389	0.1937	0.9241	19	6737
D8	0.8444	0.5420	0.9130	14	3686

focusing on various data samples marked from D1 to D8. The highest evaluation score recorded is 0.9222. It is associated with the data sample of D6. This score is achieved after 14 training epochs, suggesting that a dropout rate 6—presumably indicated by the sample identifier—leads to the best generalization performance among the tested configurations. Table III effectively illustrates how different parameters and training durations can impact the performance of a deep learning model, with particular emphasis on the evaluation scores as indicators of model effectiveness.

Table IV presents the performance outcomes for Model A when trained using a batch size of 8. It indicates that among the various trials, the one denoted as D7 yields the highest evaluation score of 0.9241. This result suggests that the settings used for dropout 7—likely corresponding to the identifier D7—result in the most effective model performance when considering the evaluation score metric compared to other dropout settings in this group. Table IV serves to compare the efficiency and accuracy of Model A under different dropout configurations while using a batch size of 8, highlighting the impact of these variables on the model’s ability to generalize and predict accurately.

Table V summarizes the performance metrics of Model A when trained with a batch size of 16 across various configurations labeled D1 through D8. The highest evaluation score within this batch size is as-

TABLE V  
MODEL A WITH BATCH SIZE 16.

No	BATCH 16				
	Accuracy	Loss	Evaluation	Epochs	Time (s)
	Last Epochs				
D1	0.9970	0.0378	0.9250	24	6468
D2	0.9919	0.0747	0.9130	19	6435
D3	0.9847	0.0831	0.9167	21	4404
D4	0.9692	0.1412	0.9194	17	3989
D5	0.9674	0.1415	0.9204	20	7127
D6	0.9514	0.1844	0.9204	21	4235
D7	0.9046	0.3228	0.9139	19	3635
D8	0.8968	0.3255	0.9259	31	8491

TABLE VI  
MODEL A WITH BATCH SIZE 32.

No	BATCH 32				
	Accuracy	Loss	Evaluation	Epochs	Time (s)
	Last Epochs				
D1	0.9894	0.1058	0.9102	26	4983
D2	0.9870	0.0873	0.9269	32	4970
D3	0.9637	0.1743	0.9222	24	6219
D4	0.9745	0.1200	0.9296	32	8153
D5	0.9694	0.1233	0.9278	37	6554
D6	0.9433	0.2183	0.9231	29	5945
D7	0.9315	0.2215	0.9296	38	6585
D8	0.7905	0.7211	0.9065	22	4256

TABLE VII  
MODEL B WITH BATCH SIZE 4.

No	BATCH 4				
	Accuracy	Loss	Evaluation	Epochs	Time (s)
	Last Epochs				
D1	0.9845	0.1357	0.6333	4	4758
D2	0.9887	0.0745	0.6750	6	2296
D3	0.9882	0.0619	0.6630	8	3355
D4	0.9826	0.0895	0.7528	10	5231
D5	0.9706	0.1896	0.7222	5	1739
D6	0.9389	0.3151	0.7204	5	2972
D7	0.9815	0.1239	0.7398	9	2535
D8	0.8660	0.5441	0.7611	7	6176

TABLE VIII  
MODEL B WITH BATCH SIZE 8.

No	BATCH 8				
	Accuracy	Loss	Evaluation	Epochs	Time (s)
	Last Epochs				
D1	0.9227	0.3086	0.8426	10	2400
D2	0.8576	0.6005	0.8713	23	3461
D3	0.9806	0.1237	0.8287	12	6207
D4	0.9079	0.3785	0.8157	12	3048
D5	0.9949	0.0600	0.8241	9	4620
D6	0.9451	0.3377	0.7333	6	4125
D7	0.9975	0.0316	0.8083	15	4244
D8	0.9722	0.1493	0.8056	14	3131

sociated with D8, which achieves an evaluation score of 0.9259. This score is notable as it is obtained after a relatively longer training period, spanning 31 epochs and taking 8491 seconds of computation time. This evaluation score represents the model's predictive performance on a validation or test dataset, with the data suggesting that the settings applied in the D8 configuration result in the best generalization capabilities of the model within the given batch size of 16.

Table VI details the results of various training sessions for Model A, utilizing a batch size 32. According to the data provided, two particular settings, corresponding to the labels D4 and D7, achieve the highest evaluation values of 0.9296. This evaluation metric is critical as it reflects the model's ability to generalize from the training data to new and unseen data. The results suggest that the parameters associated with D4 and D7, which can potentially include a dropout rate of 4 and 7, respectively, are the most effective in enhancing the model's predictive performance when using a batch size of 32.

Table VII presents data on the performance of Model B when trained with a batch size of 4. The highest score documents in the table are 0.7611 for dataset D8. This result suggests that the settings applied to dataset D8, possibly including a dropout rate represented by the dataset number, are the most conducive to achieving the best generalization performance within

the constraints of the given batch size. It also reflects a range of outcomes in terms of accuracy, loss, and training time, offering a comprehensive view of the model's training dynamics under different conditions with a batch size of 4.

Table VIII details the performance outcomes of Model B when trained with a batch size of 8 across a range of eight experimental datasets labeled D1 to D8. The evaluation score is particularly interesting as it gauges the model's predictive power on a validation set. The highest score achieved is 0.8713, associated with dataset D2. This score indicates that the training configuration used for D2, presumably representing a dropout rate of 2, is the most effective in terms of model performance within the context of a batch size of 8.

Table IX encapsulates the results of Model B when trained with a batch size of 16, as reflected in the trials labeled D1 through D8. The highest evaluation score achieved is 0.8852, corresponding to dataset D1, which is suggestive of a dropout rate of 1. This top evaluation score implies that when Model B is trained with a batch size of 16. The specified dropout setting of 1 achieves the best generalization performance on the validation dataset compared to the other dropout configurations tested. It also indicates that D1 reaches this high evaluation score after 31 epochs of training, taking 5370 seconds and suggesting a balanced trade-

TABLE IX  
MODEL B WITH BATCH SIZE 16.

No	BATCH 16				
	Accuracy	Loss	Evaluation	Epochs	Time (s)
	Last Epochs				
D1	1.0000	0.0003	0.8852	31	5370
D2	1.0000	0.0017	0.8824	22	4292
D3	1.0000	0.0005	0.8815	28	4631
D4	1.0000	0.0041	0.8843	19	4294
D5	1.0000	0.0100	0.8657	17	3438
D6	1.0000	0.0072	0.8565	20	3308
D7	0.9995	0.0196	0.8546	21	6666
D8	0.9542	0.2709	0.8019	14	4021

TABLE X  
MODEL B WITH BATCH SIZE 32.

No	BATCH 32				
	Accuracy	Loss	Evaluation	Epochs	Time (s)
	Last Epochs				
D1	1.0000	0.0040	0.9083	23	6092
D2	0.7806	0.8238	0.9148	27	4578
D3	1.0000	0.0058	0.9204	23	4769
D4	1.0000	0.0029	0.9130	29	4505
D5	0.9993	0.0168	0.9102	21	3509
D6	0.9991	0.0282	0.9111	26	6627
D7	0.9833	0.1698	0.8870	16	3244
D8	0.9845	0.1395	0.9046	22	4708

off between training duration and model performance.

Table X provides an overview of the performance metrics for Model B when trained with a batch size of 16, showcasing results from eight different configurations or data subsets labeled D1 through D8. Dataset D3, presumably corresponding to the third dropout setting, achieves the highest evaluation score of 0.9204. This score indicates the model’s effectiveness on a validation set, suggesting that the parameters used for dataset D3 are the most conducive to the model’s generalization capabilities within the batch size specified. This high evaluation score is particularly significant as it suggests that the model, with these settings, can reliably predict outcomes on unseen data.

#### IV. CONCLUSION

The design of two architectures, Model A and Model B, for a transfer learning ResNet 152 model is used for image classification of animals. Both models use all layers Conv1, Conv\_2x, Conv\_3x, Conv\_4x, and Conv\_5x, but Model B does not use freeze layers for the last three blocks of the Conv\_5x layer. Both models also included an average pooling layer, a fully connected layer, and a dropout layer before the Softmax output layer. After training, it is found that Model B with non-freeze layers achieves significantly better accuracy in classifying animal images than Model A.

The implementation of the two architectures uses a combination of batch sizes (4, 8, 16, and 32) and different dropout values (1-8). In Model A, with different batch sizes, the highest evaluation values obtained are 0.9222 for batch size 4 with dropout 6, 0.9241 for batch size 8 with dropout 7, 0.9259 for batch size 16 with dropout 1, and 0.9296 for batch size 32 with dropout 4 and dropout 7. Meanwhile, in Model B, the highest evaluation values obtained are 0.7611 for batch size 4 with dropout 8, 0.8713 for batch size 8 with dropout 2, 0.8852 for batch size 16 with dropout 1, and 0.9204 for batch size 32 with dropout 3. Implementing various batch sizes and dropouts with a value of 2 on the early stopping parameter affects the number of training iterations in both Model A and Model B. The average evaluation value in the Model A architecture does not have much impact. In contrast, in the Model B architecture, the average evaluation value has a significant effect compared to Model A.

In the research, the resulting accuracy still needs to be improved due to the large number of data classes used. Increasing the number of classes complicates the model learning process because it complicates the decision space that must be studied. It means the model must differentiate between various features to classify the input correctly. Future research is still open by using other deep learning models or modifying layers in deep learning to overcome the use of datasets with many classes.

#### ACKNOWLEDGEMENT

The research was funded by Lembaga Penelitian dan Pengabdian Masyarakat (LPPM), Institut Teknologi Telkom Purwokerto.

#### AUTHOR CONTRIBUTION

Writing—original draft, methodology S. N. S.; Conceptualization, review, F. D. A.; Formal analysis, analysis result review, U. A. All authors have read and agreed to the published version of the manuscript.

#### REFERENCES

- [1] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of tricks for image classification with convolutional neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 558–567.
- [2] C. N. Jenkins, S. L. Pimm, and L. N. Joppa, “Global patterns of terrestrial vertebrate diversity and conservation,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 28, pp. E2602–E2610, 2013.



- [3] O. Barrutia, O. Pedrera, U. Ortega-Lasuen, and J. R. Díez, "Common and threatened animal identification and conservation preferences among 6 to 12 year-old students," *Environmental Education Research*, vol. 30, no. 1, pp. 101–117, 2024.
- [4] O. A. Montesinos López, A. Montesinos López, and J. Crossa, "Fundamentals of artificial neural networks and deep learning," in *Multivariate statistical machine learning methods for genomic prediction*. Springer, 2022, pp. 379–425.
- [5] M. Salvi, U. R. Acharya, F. Molinari, and K. M. Meiburger, "The impact of pre- and post-image processing techniques on deep learning frameworks: A comprehensive review for digital pathology image analysis," *Computers in Biology and Medicine*, vol. 128, pp. 1–24, 2021.
- [6] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li, "Development of convolutional neural network and its application in image classification: A survey," *Optical Engineering*, vol. 58, no. 4, pp. 040 901–040 901, 2019.
- [7] A. Mahbod, G. Schaefer, I. Ellinger, R. Ecker, A. Pitiot, and C. Wang, "Fusing fine-tuned deep features for skin lesion classification," *Computerized Medical Imaging and Graphics*, vol. 71, pp. 19–29, 2019.
- [8] G. Habib and S. Qureshi, "Optimization and acceleration of convolutional neural networks: A survey," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 4244–4268, 2022.
- [9] R. R. E. Prasetyo and M. Ichwan, "Perbandingan metode Deep Residual Network 50 dan Deep Residual Network 152 untuk deteksi penyakit pneumonia pada manusia," *MIND (Multimedia Artificial Intelligent Networking Database) Journal*, vol. 6, no. 2, pp. 168–182, 2021.
- [10] W. Xu and R. S. Cloutier, "A facial expression recognizer using modified ResNet-152," *EAI Endorsed Transactions on Internet of Things*, vol. 7, no. 28, pp. 1–9, 2022.
- [11] A. Berk, G. Ozturan, D. Maberley, Ö. Yılmaz, and I. Oruc, "Learning from few examples: Classifying sex from retinal images," *Journal of Vision*, vol. 20, no. 11, pp. 255–255, 2020.
- [12] A. Berk, G. Ozturan, P. Delavari, D. Maberley, Ö. Yılmaz, and I. Oruc, "Learning from small data: Classifying sex from retinal images via deep learning," *PLOS ONE*, vol. 18, no. 8, pp. 1–24, 2023.
- [13] G. A. Anwar and D. Rimirasih, "Klasifikasi citra genus Panthera menggunakan metode Convolutional Neural Network (CNN)," *Jurnal Ilmiah Informatika Komputer*, vol. 24, no. 3, pp. 220–228, 2020.
- [14] F. D. Adhinata, N. G. Ramadhan, and A. Jayadi, "Densenet201 model for robust detection on incorrect use of mask," in *Proceeding of the 3<sup>rd</sup> International Conference on Electronics, Biomedical Engineering, and Health Informatics: ICE-BEHI 2022, 5–6 October, Surabaya, Indonesia*. Springer, 2023, pp. 251–263.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [16] D. Ji, Z. Zhang, Y. Zhao, and Q. Zhao, "Research on classification of COVID-19 chest x-ray image modal feature fusion based on deep learning," *Journal of Healthcare Engineering*, vol. 2021, pp. 1–12, 2021.
- [17] M. Krichen, "Convolutional neural networks: A survey," *Computers*, vol. 12, no. 8, pp. 1–41, 2023.
- [18] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," *Advances in Neural Information Processing Systems*, vol. 31, pp. 1–12, 2018.
- [19] M. M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," *Computation*, vol. 11, no. 3, pp. 1–23, 2023.
- [20] I. Salehin and D. K. Kang, "A review on dropout regularization approaches for deep neural networks within the scholarly domain," *Electronics*, vol. 12, no. 14, pp. 1–23, 2023.
- [21] A. A. M. S. Ibrahim and J. R. Tapamo, "Transfer learning-based approach using new convolutional neural network classifier for steel surface defects classification," *Scientific African*, vol. 23, pp. 1–11, 2024.
- [22] R. A. Pangestu, B. Rahmat, and F. T. Anggraeny, "Implementasi algoritma CNN untuk klasifikasi citra lahan dan perhitungan luas," *Jurnal Informatika dan Sistem Informasi*, vol. 1, no. 1, pp. 166–174, 2020.
- [23] R. A. Charisma and F. D. Adhinata, "Transfer learning with Densenet201 architecture model for potato leaf disease classification," in *2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE)*. IEEE, 2023, pp. 738–743.