International Journal of Computer Applications (0975 - 8887)
Volume 186 - No.61, January 2025

MobileNetV2: Transfer Learning for Elephant Detection

Samay Sawal
Intern Data Scientist
Valiance Solutions
Noida, India

Shailendra Singh Kathait
Co-Founder and Chief Data Scientist
Valiance Solutions
Noida, India

ABSTRACT

This study presents a very new and innovative approach to classify the elephants using a deep learning framework [2] which is built on transfer learning and data augmentation techniques. By using the features of MobileNetV2 as the base model, followed by different layers, the model achieved a high accuracy in classifying between images of elephants and other objects. This paper explains in detail about the end-to-end process, including dataset preparation, pre-processing, model architecture, and evaluation metrics. The results indicated the effectiveness of the proposed model in obtaining a high classification accuracy with a robust generalization across the training and validation datasets.

General Terms

Elephant Classification, Animals, Forest, Animal Identification, Pattern Recognition, Wildlife Preservation.

Keywords

Elephant, Classification, Deep Learning, MobileNetV2, Transfer Learning, Data Augmentation.

1. INTRODUCTION

The accurate classification [8] of wildlife is very important for correct ecological monitoring and efforts for conservation. Among all the wildlife species, elephants play one of the most important role in maintaining ecological balance in various different ways. Their activities such as seed dispersal and habitat modification influence many other species which makes their presence and health an indicator to judge the ecological well being. The classical methods for identifying and classifying elephants often rely on manual observation and recording. These methods are not only labor intensive but also are prone to many human errors. These limitations creates a need for reliable and automated solutions to improve precision and efficiency in elephant classification.

For wildlife managers, in particular, the right classification of elephants is of great importance. This is because, they make it possible for the managers to acquire accurate data within a short time. Detailed information about elephants helps the wildlife managers in implementing better resources and designing more efficient interventions. Furthermore, because of human-wildlife conflict, poaching, and habitat loss, elephant populations are often vulnerable and prone to effective monitoring and classification systems are crucial for the conservation of such groups and the species in the long run.

In the present study, transfer learning is integrated with MobileNetV2 architecture [11] to fulfill the necessity of introducing new approaches in the wildlife classification. A novel lightweight CNN architecture called MobileNetV2 is known for its ease of image classification while having lesser computations than traditional architectures. Using MobileNetV2 features and a visual-based dataset, a model that can effectively classify elephants is presented. New approaches in deep learning classification often require high computational facilities turning out to be a disadvantage for the models. Higher efficiency in monitoring and tracking wildlife populations like elephants are enabled through below methodology, their habitats and also intrusion detection systems.

2. METHODOLOGY

2.1 Dataset Preparation

The images captured by the cameras are taken from the BRT tiger reserve, which are eventually stored in cloud buckets, labeled with information such as camera number, time and location from where it was captured. Although for the training of the model we only require the class of the image. information. The dataset was structured into two classes: ”elephants” and ”others.” All images were gathered in various lighting conditions and poses to ensure robust training. The integrity of class-specific directories was checked, ensuring balanced representation across categories.

2.2 Data Augmentation and Pre-processing

Pre-processing normalizes pixel values to a scale of [0, 1] to ensure consistency across all datasets. The ImageDataGenerator class is used for this purpose.

(1) Training: rescale=1./255 is applied to scale image pixel values
(2) Validation and Test: Rescaling ensures compatibility with the trained model

For Data Augmentation a series of transformation were carried out to improve the diversity in the dataset:

(1) Rescaling: Pixel values are normalized to the range [0, 1]
(2) Rotation: Images are randomly rotated up to 40 degrees

(3) Shifting: Random horizontal and vertical shifts are applied up to 20 % of the image dimensions
(4) Shearing: Images undergo random shearing transformations
(5) Zooming: Random zooms up to 20 % are applied
(6) Flipping: Horizontal flipping is performed
(7) Fill Mode: Missing pixels from transformations are filled using the nearest method

These augmentations simulate real-world variations and enhance the robustness of the model.

Generators are configured to load and pre-process images dynamically during training, validation, and testing:

(1) Training Generator: Combines augmentation and pre-processing, shuffling data for each epoch to ensure model exposure to diverse transformations
(2) Validation Generator: Performs pre-processing without augmentation, providing consistent data for validation
(3) Test Generator: Similar to the validation generator, ensuring unbiased evaluation

The configuration parameter included: target_size: Standardized image dimensions, batch_size: Number of images per batch, class_mode: Specifies binary classification, shuffle: Determines whether data is randomized.

This research shows how significant data augmentation and pre-processing are in a image classification tasks. The use of ImageDataGenerator provides a flexible and scalable solution for enhancing datasets and optimizing neural network performance

The data was split into training (60%), validation (20%) and test (20%) subsets using **ImageDataGenerator**. The dataset was prepared for binary classification, with class indices generated for mapping.

2.3 Dataset Images

The images have been procured from BRT tiger Reserve. The features like time, camera number and location have been removed from the image for better visualization here.

Below are few of the images of both the classes that were being used in the training process:

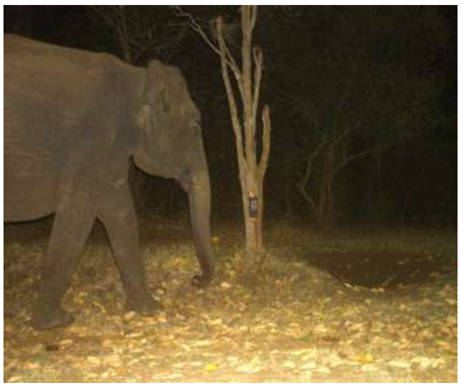
Fig 1. Image Class = Elephant

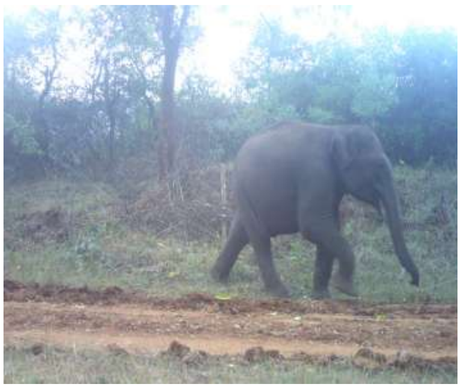
Fig 2. Image Class = Elephant

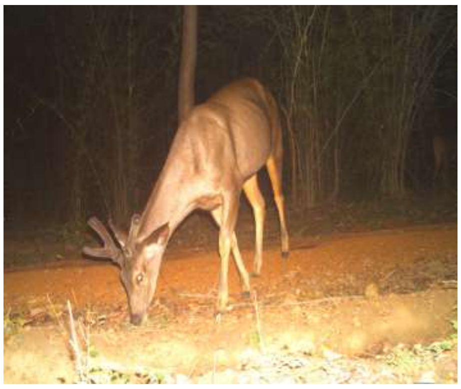
Fig 3. Image Class = Others

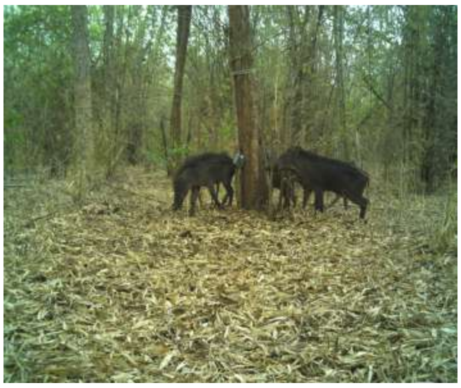
Fig 4. Image Class = Others

2.4 MobileNetV2

MobileNetV2 [7] is a lightweight and efficient convolutional neural network [10] architecture designed for mobile and embedded vision applications. It is a successor to the original MobileNet and improves upon it by introducing key architectural innovations for enhanced performance and reduced computational complexity. MobileNetV2 is widely used for tasks such as image classification, object detection, and semantic segmentation.

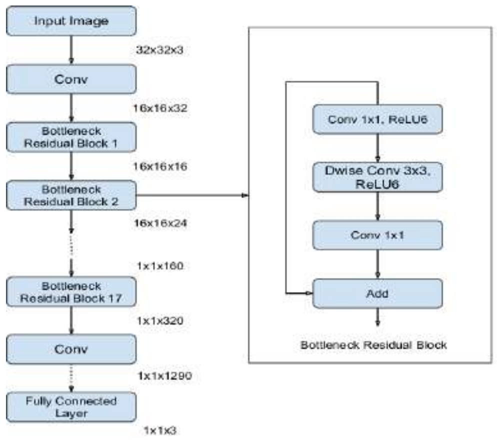

Fig 5. MobileNetV2 Architecture

Step wise process in the model:

(1) Input Image: The input to the model is an image of size 32×32×3, where 32×32 represents the spatial resolution, and 3 denotes the RGB channels
(2) Initial Convolutional Layer: A convolutional layer applies a filter to extract initial low-level features, such as edges and textures. The output after this layer has a size of 16×16×32, where 32 is the number of feature maps generated by the filters
(3) Bottleneck Residual Blocks: The main building block of this architecture is the Bottleneck Residual Block. These blocks are efficient for feature extraction and form the backbone of the network
(4) Transition Layers: After the final bottleneck block, a convolutional layer further adjusts the dimensionality of features. The output size is reduced to 1×1×1290, representing highly condensed spatial information
(5) Fully Connected Layer: The final stage is a fully connected layer that maps the features into the output class probabilities. The output size is 1×1×3, corresponding to a classification task with 3 output classes

Bottleneck Residual Block (Detailed View):
Inside each block:

(1) Point wise Convolution (1×1 Conv, ReLU6): Reduces or expands the number of channels, adjusting feature dimensionality
(2) Depth wise Separable Convolution (3×3, ReLU6): This efficiently applies convolutions to each channel separately, reducing the computational cost
(3) Point wise Convolution (1×1 Conv): This restores the channel dimensions as required by the architecture
(4) Skip Connection: This adds the input of the block to its output (if the dimensions match), enabling the gradient flow and improving model optimization

The output dimensions of these blocks may vary, with each block thus transforming the spatial and channel dimensions.

2.5 Transfer Learning

Transfer Learning [1] is a machine learning technique where a model trained on one task is adapted to perform a different but related task. It uses the pre-trained models those are trained on large datasets for general tasks (like image recognition or natural language processing)—and fine tunes them to solve specific problems with smaller datasets.

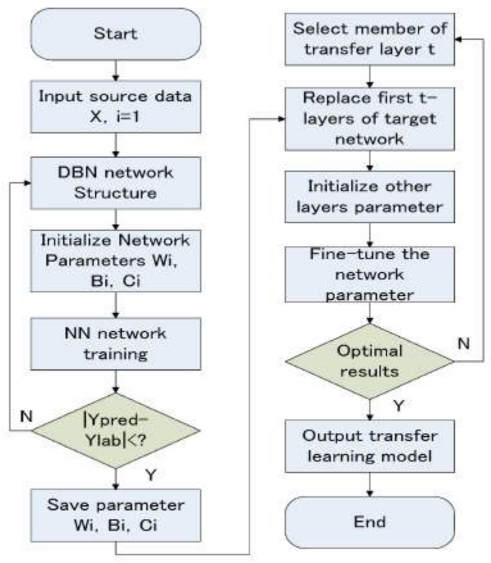
Start
Input source data X, i=1
DBN network Structure
Initialize Network Parameters Wi, Bi, Ci
NN network training
|Ypred−Ylab|<?
N
Y
Save parameter Wi, Bi, Ci
Select member of transfer layer t
Replace first t−layers of target network
Initialize other layers parameter
Fine−tune the network parameter
Optimal results
N
Y
Output transfer learning model
End

Fig 6. Flowchart for Transfer Learning approach

This methodology highlights the integration of DBN training and transfer learning [9] to improve neural network performance in new domains. By transferring knowledge from a pre-trained model, the framework enables faster convergence and better generalization for the target task, making it a valuable approach in scenarios with limited labeled data.

(1) Start: The process begins by initializing the framework for data input and network training
(2) Input Source Data: The source data, represented as X, is fed into the system, starting with the first dataset (i=1)
(3) DBN Network Structure Initialization: A deep belief network (DBN) is constructed with an appropriate architecture suitable for the source data
(4) Initialize Network Parameters: The initial parameters of the DBN network are set, including weights (W_i), biases (B_i), and other connection weights (C_i)
(5) Neural Network (NN) Training: The DBN undergoes standard training to optimize its parameters for the given data
(6) Evaluate Performance: The prediction error, computed as $|Y_{pred} - Y_{lab}|$, is checked against a predefined threshold. If the error is within the acceptable range, the model parameters (W_i, B_i, C_i) are saved otherwise the process loops back for further refinement of the DBN
(7) Transfer Learning Phase: First select a specific layer (t) from the source network to transfer to a target network. Then the first t-layers of the target network are replaced with the corresponding layers from the trained DBN and the parameters for the other layers are also initialized
(8) Fine-tune Network Parameters: The target network is fine-tuned using transfer learning techniques to optimize its performance for the new task or dataset
(9) Evaluate Transfer Learning Results and output results

2.6 Experiments Discussion

Before finalizing the model architecture various Deep Learning [11] experiments were carried out to find the best architecture for elephant classification. Following are the experiments done:

(1) **Training a CNN form scratch:** A Convolutional Neural Network (CNN) [12] was built from scratch without any pre-trained models. Architecture involved three 2D Convolutional layers with three max pooling layers and flattening of the layers. It also involved a dense layer with 128 neurons, a dropout layer and a output layer which used a sigmoid activation function. This is a classical image classification architecture which was employed to classify elephants.
This architecture was discarded since the it gave the test accuracy of 86.54 % and validation loss was 0.3093. These performance metric are lower than the final approach and thus this model was discarded for elephant classification.

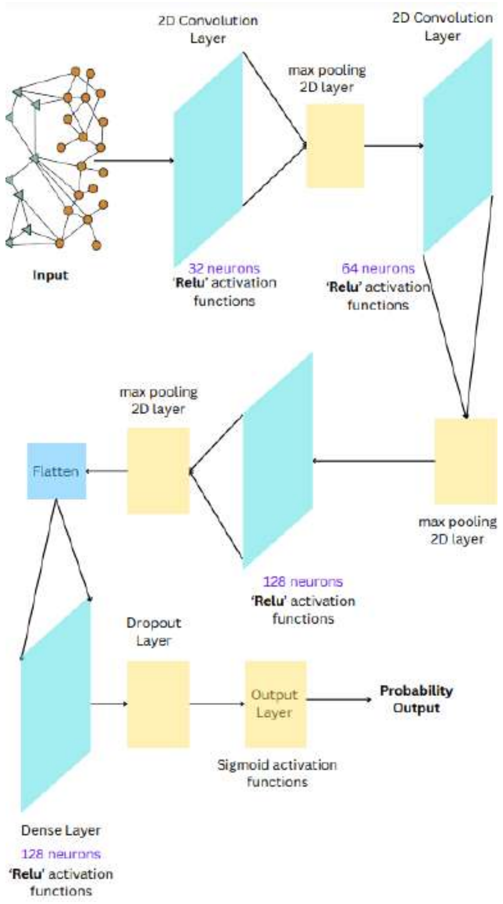

Fig 7. CNN Architecture employed for elephant classification

(2) **Pre-trained VGG16 Architecture:** The experiments with the VGG16 [13] architecture provided essential insights into the limitations of using deep, pre-trained networks for smaller datasets. While VGG16's robust feature extraction capabilities were known but the model's weakness to overfit, its computational overhead, and its inefficiency for our task ultimately led to shifting to a different architecture. These findings led to the need for a more tailored architecture capable of better generalization and faster convergence, which helped us to select the final model outlined in the next section of this paper.
VGG16 (Base Model) - GlobalAveragePooling2D Layer - Dense Layer - Output Layer. This was the architecture for this model. The performance metric is as follows:

(a) Validation Loss: 0.5855
(b) Validation Accuracy: 85.45 %

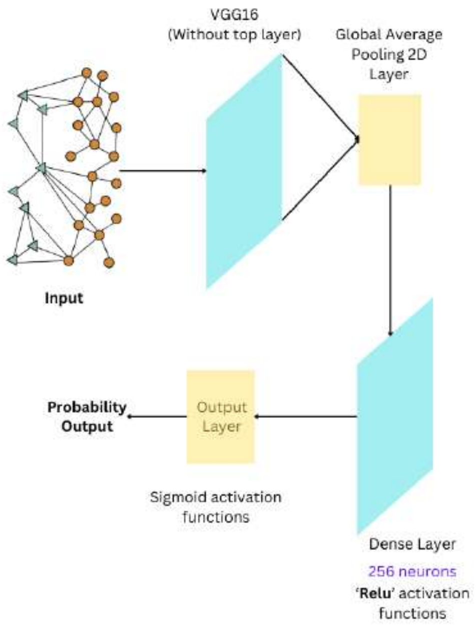
VGG16 (Without top layer)
Global Average Pooling 2D Layer
Input
Probability Output
Output Layer
Sigmoid activation functions
Dense Layer
256 neurons
'Relu' activation functions

Fig 8. VGG16 Architecture employed for elephant classification

2.7 Model Architecture

A transfer learning-based approach [1] was used in the approach, leveraging MobileNetV2 as the feature extractor. The Base Model that is MobileNetV2 pre-trained on ImageNet [6] with the top classification layer removed was used.
In this architecture MobileNetV2 [7] as seen above was initialized with pre-trained ImageNet weights, excluding the top layers and it was Configured to accept images of size 224x224 with three color channels. The base model's layers were frozen (trainable=False) to retain the pre-learned features.
MobileNetV2 was more favourable for our architecture as,

(1) MobileNetV2 is pre-trained on the ImageNet dataset, which contains over a million images across 1,000 classes. This extensive pre-training enables the model to learn rich and diverse feature representations that are transferable to other image classification tasks, including elephant classification
(2) It employs depthwise separable convolutions, significantly reducing the number of parameters and computational load compared to traditional convolutional networks. This efficiency is beneficial for handling large datasets and for deployment on devices with limited computational resources
(3) The architecture introduces inverted residual blocks with linear bottlenecks, enhancing the model's ability to capture fine-grained features while maintaining a lightweight structure. This design choice contributes to improved performance without incurring additional computational costs

2.8 Custom Layers in Model Architecture

There were few custom layers that were added in the model architecture to improve the classification ability. This includes the following alongwith the advantage it gave for the classification purpose-

(1) Global Average Pooling: Incorporating a GlobalAveragePooling2D layer reduces the spatial dimensions of the feature maps output by the base model, converting them into a single vector per feature map. This operation effectively summarizes the presence of features across the entire image, which is beneficial for classification tasks
(2) Fully Connected Layer with ReLU Activation: Adding a dense layer with 128 units and ReLU activation introduces non-linearity and allows the model to learn complex patterns and feature combinations specific to elephant images. This layer acts as a bridge between the generic features learned by the base model and the specific patterns required for accurate classification
(3) Dropout Layer: Implementing a Dropout layer with a rate of 0.5 provides regularization by randomly setting input units to zero during training. This prevents co-adaptation of features and reduces the likelihood of over-fitting, enhancing the model's generalization capabilities
(4) Output Layer with Sigmoid Activation: The final dense layer uses a sigmoid activation function, appropriate for binary classification or multi-label classification scenarios. It outputs probabilities that an input image belongs to each class, facilitating threshold-based decision-making

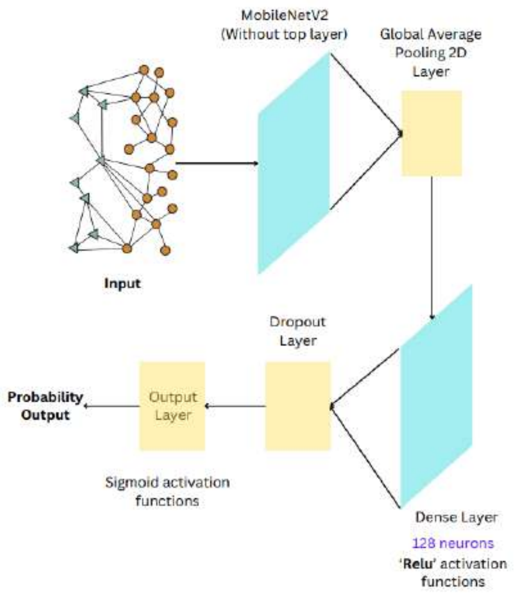

Fig 9. Final Model Architecture

2.9 Training and Validation

The model was trained using the Adam optimizer and binary cross-entropy loss function, ensuring compatibility with binary classification tasks. The model is trained for up to 20 epochs with a batch size of 32. The training process involves iterating over the training data, updating weights based on the loss, and validating on the reserved subset.

Early Stopping was employed which monitors the validation loss and stops training if no improvement is observed over five consecutive epochs, restoring the best weights and model checkpoint was set up as it saves the best model based on validation loss during training.

2.10 Evaluation Metrics

Key metrics included:

(1) Accuracy: Overall, correct predictions
(2) Loss: Binary cross-entropy loss
(3) Learning Curves: Visualization of training and validation accuracy and loss over epochs

Post-training, the model's performance is evaluated on the validation set using metrics such as loss, accuracy, confusion matrix, and classification report as mentioned above. Visualization of training history provides insights into the model's learning dynamics.

3. RESULTS AND DISCUSSION

3.1 Performance Metrics

Before training, the dataset was inspected to ensure the presence of images in each class. The script verifies the existence of class directories and counts the number of images:

```
Found X images in 'elephants'
Found Y images in 'others'
```

(1) X: Number of Images in the "elephant" Class
(2) Y: Number of Images in the "others" Class

The final model acheievd the following accuracy and loss:

(1) Test Dataset Loss: 0.0467
(2) Test Accuracy: 98.31 %

Convergence was indicated by the model's learning curves which showed an increase in accuracy as well as a decrease in loss on both training and validation datasets. These precision and recall values show that the model can properly identify both classes with very few misclassifications. The pre-trained base model-MobileNetV2 [7] came in the middle between computational cost and accuracy. The pre-trained weights from ImageNet [6] made it possible for the model to use learned features, saving time during training and improving convergence speed.

Data augmentation [3] played a crucial role in enhancing the model's generalization capabilities. By introducing variability in the training data, the model became more robust to changes in orientation, scale, and lighting conditions, reducing overfitting and improving performance on unseen data.

3.2 Inference Testing

Inference testing on unseen data validated the model's robustness across varied environmental conditions, including lighting and backgrounds. The model demonstrated high accuracy in distinguishing elephants from non-elephant images, emphasizing its utility for real-world applications.

A threshold was set at the probability of 0.5 at which a Elephant class will be predicted. With enough individual images of these Elephants, this architecture can be employed to a bigger dataset and the model can be even more robust.

Below are the confusion matrices (the confusion matrix offers a comprehensive view of the model's classification performance, allowing for a nuanced understanding of its strengths and weaknesses across different classes. It serves as a vital tool for improving model performance through targeted interventions such as retraining, hyperparameter tuning, or adjustments to the dataset) for all the three architectures we have experimented and it shows clearly that performance of MobileNetV2 surpasses other architectures significantly on unseen data.

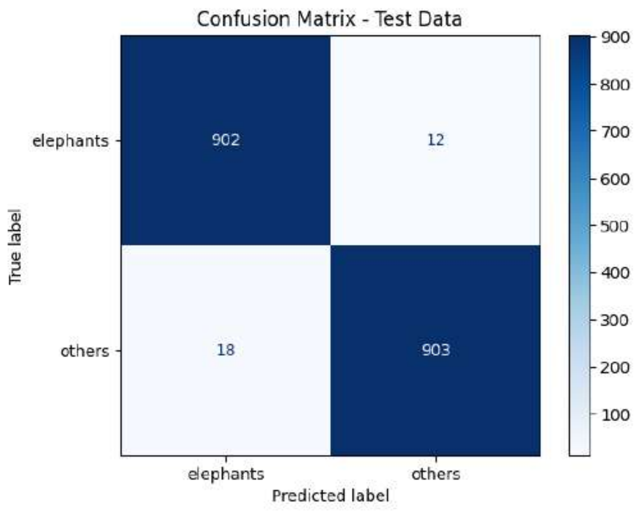

Fig 10. Confusion Matrix for MobileNetV2 Architecture

The final model acheievd the following accuracy and loss:

(1) Test Dataset Loss: 0.0467
(2) Test Dataset Accuracy: 98.31 %

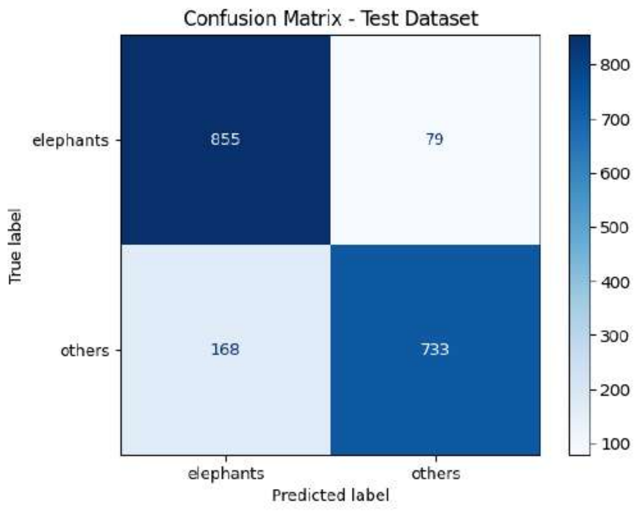

Fig 11. Confusion Matrix for CNN Architecture

The final model acheievd the following accuracy and loss:

(1) Test Dataset Loss: 0.3093
(2) Test Dataset Accuracy: 86.54 %

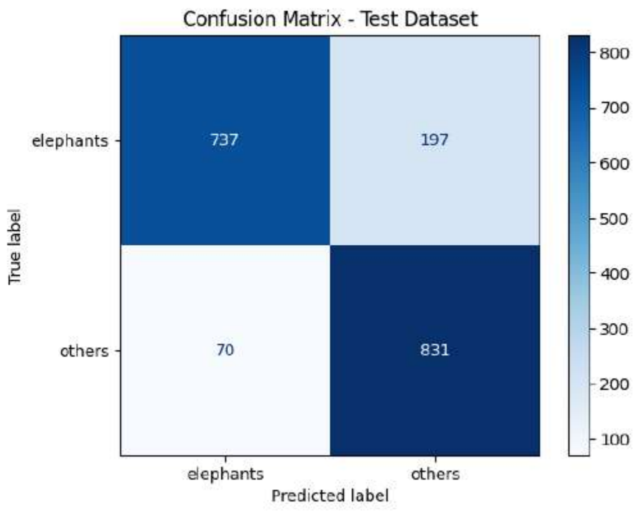

Fig 12. Confusion Matrix for VGG16 Architecture

The final model acheievd the following accuracy and loss:

(1) Test Dataset Loss: 0.5855
(2) Test Dataset Accuracy: 85.45 %

Recommended Model:

Based on the performance metrics and experiments discussed in the paper the MobileNetV2 architecture is recommended as the best model for elephant classification due to its lightweight nature, computational efficiency, and high accuracy. It achieved 98.31 % accuracy on the test dataset, much better than the other architectures like CNN (86.54 %) and VGG16 (85.45 %). Its pre-trained ImageNet weights and the use of advanced techniques such as inverted residual blocks and depthwise separable convolutions made it highly effective for this task of elephant classification.

3.3 Conclusion and Future Work

This test shows that deep learning models, especially MobileNetV2 [7], are a really powerful answer to wildlife monitoring and conservation, especially in their ability to identity elephants from anything else. Applying transfer learning [1], data augmentation [3], and custom layers helps improve the generalization and classification [8] functions of the model. The results stress further the applicability of deep learning models to wildlife conservation and other real-world applications. The work could be built on in future studies to further improve model performance, extend into multi-class classification, and find real-world applications.

Future work will focus on:

(1) Individual Identification of Elephants:
    —Implement a system to identify individual elephants using unique physical traits such as tusk shape, ear patterns, and scars
    —Incorporate advanced models like EfficientNet or fine-tuned transformers for feature-rich classification
    —Utilize Siamese or triplet networks for similarity-based learning to distinguish individuals

(2) Size Estimation:
—Extend the model to estimate the size of elephants by incorporating depth information or using object-detection models like YOLO or Faster R-CNN
—Train with datasets annotated with size-related metadata

(3) Augmentation of Dataset: Include more varied environmental conditions and additional labeled data, specifically focusing on individual features

(4) Real-time deployment:
—Optimize the model for edge devices for real-time monitoring
—Explore TinyML for reducing computational requirements further without sacrificing accuracy

(5) Incorporation of Advanced Augmentation: Integrate domain-specific augmentations like brightness adjustment and partial occlusion to improve robustness

(6) Multi-Class Classification: Extend the model to identify other species or classify elephant subcategories

The outlined steps would make the model even more robust and practical for field deployment, helping conservationists monitor and analyze elephant populations effectively.

4. REFERENCES

[1] M. H. Salem, Y. Li, and Z. Liu. ”Transfer Learning on EfficientNet for Maritime Visible Image Classification,” 2022 7th International Conference on Signal and Image Processing (ICSIP), Suzhou, China, 2022, pp. 514-520, doi: 10.1109/ICSIP55141.2022.9887050.

[2] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021).

[3] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019).

[4] Tammina, Srikanth. (2019). Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. International Journal of Scientific and Research Publications (IJSRP). 9. p9420. 10.29322/IJSRP.9.10. 2019.p9420.

[5] Tan, M. and Le, Q., 2021, July. Efficientnetv2: Smaller models and faster training. In International conference on machine learning (pp. 10096-10106). PMLR.

[6] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), pp.84-90.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks, doi: 10.48550/arXiv.1801.04381.

[8] Mothukuri Sujith, Shailendra Singh Kathait, Piyush Dhuliya Valiance Analytics Private Limited. Artificial Intelligence for Human-Animal Conflict Mitigation: Image Classification and Human Tracking in Tadoba Andhari Tiger Reserve. *”International Conference on Human-Elephant Conflict Management”*

[9] Shailendra Singh Kathait, Vaibhav Singh, Ashish Kumar (2024) Valiance Analytics Private Limited. Individual Tiger Identification using Transfer Learning. *International Journal of Computer Applications (0975 – 8887) Volume 186 – No.11, March 2024*

[10] Kartikya Gupta, Vaibhav Sharma, Shailendra Singh Kathait (2024) Valiance Analytics Private Limited. Smart Screening: Non-Invasive Detection of Severe Neonatal Jaundice using Computer Vision and Deep Learning. *International Journal of Computer Applications (0975 – 8887) Volume 186 – No.35, August 2024*

[11] Shailendra Singh Kathait, Ashish Kumar, Piyush Dhuliya, Ikshu Chauhan (2024) Valiance Analytics Private Limited. Deep Learning-based Model for Wildlife Species Classification. *International Journal of Computer Applications (0975 – 8887) Volume 186 – No.1, January 2024*

[12] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[13] Karen Simonyan, Andrew Zisserman (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. doi: arXiv.1409.1556