Jurnal Masyarakat Informatika, 16(1), 2025
DOI:10.14710/jmasif.16.1.72957
ISSN: 2777-0648

A Comparative Analysis of Convolutional Neural Network (CNN): MobileNetV2 and Xception for Butterfly Species Classification

Mehta Pradnyatama[1)], Christy Atika Sari*[1)], Eko Hari Rachmawanto[1)], and Hussain Md Mehedul Islam[2)]

[1)]*Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang, Indonesia*
[2)]*Software Engineer, The Mathworks, Inc., United States*
[*] *Corresponding author: christy.atika.sari@dsn.dinus.ac.id*

Abstract

This study aims to compare the effectiveness and efficiency of two convolutional neural network architectures, MobileNetV2 and Xception, for automated butterfly species classification. As biodiversity monitoring gains significance, effective species identification technologies are crucial for conservation. The research utilized a dataset of 100 butterfly species with 12,594 training images and 1,000 validation and test images. Transfer learning with pre-trained ImageNet weights was implemented, and both models were enhanced with custom classification layers. Data augmentation and class weighting mitigated dataset imbalance issues. Experimental results show Xception attained 93.40% test accuracy compared to MobileNetV2's 93.20%. These high accuracy rates were achieved through effective transfer learning that preserved general feature extraction capabilities, comprehensive class balancing techniques, and carefully tailored learning rate strategies for each architecture. Despite minimal performance difference, MobileNetV2 offers significant computational efficiency advantages with 4.15M parameters compared to Xception's 25.27M, while Xception provides marginally better classification. This study contributes to entomological research and highlights trade-offs between model complexity and performance in fine-grained classification tasks, supporting implementation decisions for butterfly identification systems in practical applications.

Keywords : *Classification, Convolutional Neural Network, MobileNetV2, Xception, Butterfly*

1 Introduction

The classification of butterfly species was essential for biodiversity monitoring, ecological study, and global conservation initiatives. With the acceleration of climate change and habitat destruction, monitoring butterfly populations has become crucial for evaluating ecosystem health [1]. Conventional butterfly classification techniques depend extensively on expert knowledge and intensive for resulting in considerable obstacles in extensive biodiversity research [2]. The advent of deep learning methodologies, especially *Convolutional Neural Networks (CNNs)*, has transformed automated visual recognition tasks, providing effective solutions for species identification issues [3].

Recent advancements in *CNN* architectures have yielded robust instruments for automated butterfly classification. where pre-trained models are adapted for specific domains, has become the standard approach for biological image classification tasks where limited training data is available [2]. Among the diverse *CNN* architectures, lightweight models such as *MobileNetV2* provide computational efficiency suitable for deployment in resource-limited settings, whereas more intricate

Received: 05 May 2025 ; Revised: 25 May 2025; Accepted: 29 May 2025; Published: 31 May 2025

architectures like *Xception* may yield superior accuracy at the expense of heightened computaxtional requirements [4].

Numerous studies have explored automated butterfly classification with deep learning methodologies. For example, *MobileNetV2* has exhibited impressive performance in entomological applications, with up to 94.6% accuracy in butterfly species recognition tasks and 86% accuracy in a classification including 75 species [5]. *Xception*, utilizing depthwise separable convolutions and significant architectural depth, has attained 80.58% accuracy at the species level and up to 98.02% at the order level in insect classification tasks [4]. Alternative *CNN*-based methodologies, such as modified *InceptionV3* and hybrid models, have demonstrated elevated accuracy, with several studies attaining over 98% accuracy on extensive butterfly datasets [6].

Despite these advances, direct comparative analyses of *MobileNetV2* and *Xception* for butterfly species classification are scarce, especially concerning extensive, heterogeneous datasets. While recent studies have made valuable contributions, such as Kaur's single-model optimization achieving 94.6% accuracy with MobileNetV2 for 75 butterfly species and Santhiya et al.'s ensemble approach combining VGG19 and Xception for general insect classification across mixed taxonomic orders, several critical gaps remain in the literature [4][5]. These existing works either focus on individual model optimization without systematic architectural comparison, address mixed insect taxonomies rather than specialized butterfly classification, or lack comprehensive statistical validation of performance differences.

This study addresses these limitations through a comprehensive comparative framework that advances the field in multiple dimensions. First, we provide rigorous statistical validation of architectural performance differences using McNemar's test and paired t-test analysis [7], which represents statistical rigor absent in previous comparative studies. Second, our research offers the first systematic computational efficiency benchmarking for butterfly classification, measuring inference time and memory usage critical for deployment decisions in resource-constrained field applications. Third, we conduct detailed per-class performance evaluation across 100 butterfly species, revealing species-specific classification challenges that provide deeper insights than aggregate accuracy metrics alone.

Our approach further contributes through domain-specialized analysis, focusing exclusively on large-scale butterfly classification rather than mixed insect taxonomies, enabling more precise understanding of lepidopteran visual recognition challenges [8]. Additionally, we implement systematic class imbalance mitigation through inverse frequency weighting, addressing a key practical challenge unaddressed in existing comparative studies. Our comprehensive training dynamics analysis, including interpretation of validation loss patterns and convergence behavior, provides interpretable insights into deep learning optimization for biological image classification [9]. Finally, we establish a deployment-oriented trade-off framework that balances accuracy and computational efficiency, offering practical guidance for real-world conservation applications rather than pursuing accuracy maximization alone.

Based on these identified gaps, this study implements a systematic comparison of *MobileNetV2* and *Xception* architectures for butterfly species classification using a comprehensive dataset of 100 distinct species, with findings intended to guide practical deployment decisions in conservation and entomological research applications [2][6].

2 Research Methods

This research performed a systematic comparison of the MobileNetV2 and Xception architectures for the classification of butterfly species utilizing an eight-stage methodological methodology. The methodology combined standard deep learning protocols with bespoke tweaks to tackle the specific issues of butterfly image processing. Figure 1 illustrates the research framework, detailing the methodological path from dataset preparation to performance evaluation. This systematic method enabled uniform assessment of both architectures throughout all research phases, guaranteeing fair comparison of their classification capabilities.

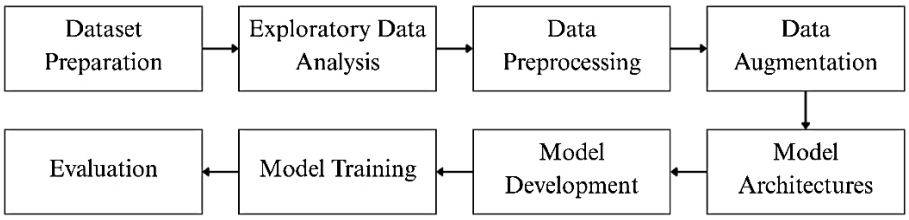

Figure 1 Research Methodology Workflow

2.1 Dataset Preparation

The dataset used in this study was sourced from Kaggle's public repository, which provides a comprehensive collection of butterfly images across numerous species. The selection of this comprehensive 100-species butterfly dataset was strategically motivated by several factors that distinguish it from alternative datasets commonly used in lepidopteran classification research. While existing studies have typically employed smaller taxonomic scopes ranging from 10-75 species, our dataset provides enhanced taxonomic diversity essential for robust architectural comparison and generalization assessment. This collection offers superior class representation compared to fragmented datasets, with standardized image quality and consistent annotation protocols that ensure reproducibility across different computational environments. The dataset's public availability through Kaggle's repository facilitates research reproducibility and enables direct comparison with future studies, addressing a critical gap in butterfly classification research where dataset fragmentation has hindered comprehensive architectural evaluations. Furthermore, the 100-species scope provides sufficient complexity to reveal meaningful performance differences between lightweight and complex CNN architectures while maintaining manageable computational requirements for systematic comparison. In the preparation and organization of datasets for classifying 100 butterfly species, this study structured the data into hierarchical directories comprising training, validation, and testing folders, each containing images appropriately partitioned for model development and evaluation purposes [10]. Image standardization to 224×224×3 pixel format represented a critical preprocessing step to ensure dimensional consistency for convolutional neural network inputs, as implemented in our butterfly classification methodology [11]. Our dataset distribution strategy allocated 12,594 images for training, 500 for validation, and 500 for testing, contributing significantly to maintaining data balance and enhancing model performance. Directory validation and verification protocols were systematically implemented to confirm proper data segregation across training,

validation, and testing subsets while preventing cross-contamination between partitions—a standard practice in butterfly classification dataset development [11]. Figure 2 shows a graph of the distribution of the datasets for training, validation, and testing, showing how many images are in each group. The bar graphs show how many images were given to each class for training, validation, and testing.

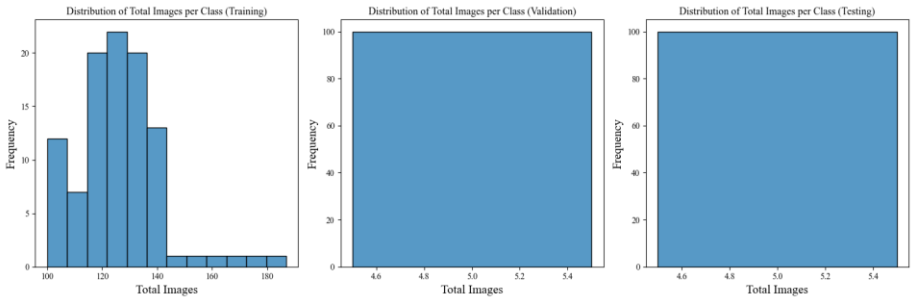

Figure 2 Dataset Distribution

Figure 3 illustrates representative images from the dataset, showcasing butterflies in the training, validation, and testing subsets. Each row displays images from distinct butterfly species, accompanied with labels for the train, validation, and test. This graphical depiction aids in comprehending the allocation of photos among various categories and dataset segments.

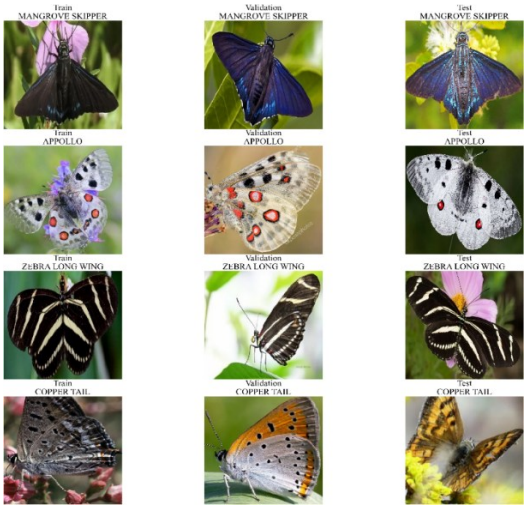

Figure 3 Dataset Samples

2.2 Exploratory Data Analysis

This study Exploratory Data Analysis (EDA) revealed moderate class imbalance across the butterfly dataset, with statistical analysis showing training samples ranging from 100 to 187 images per species (mean=125.94, SD=15.26), while validation and test sets maintained consistent 5 samples per class. Specifically, "*Sixspot Burnet Moth*" emerged as the minority class with 100 samples and "*Mourning Cloak*" as the majority class with 187 samples, necessitating appropriate mitigation strategies [12]. Class weight calculations were implemented inversely proportional to sample counts, assigning weights of 1.0 for classes with 100 samples and 0.5348 for classes with 187 samples, effectively addressing potential classification bias [13]. Visualization through histograms and bar

charts facilitated class distribution analysis and informed subsequent data augmentation techniques. This approach aligns with research demonstrating that appropriate handling of class imbalance significantly improves model performance in fine-grained image classification tasks [14], particularly important in biological datasets where certain species are frequently underrepresented [15]. Quantitative analysis of class distribution provided essential guidance for implementing effective preprocessing strategies to ensure balanced model training across all butterfly species. Figure 4 provides a visual representation of the class distribution within the dataset, offering insights into the classes with the most and the fewest samples. The bar charts below depict the top 10 classes with the highest number of training images, as well as the 10 classes with the fewest images. This visualization highlights the disparities in sample distribution across the butterfly species and emphasizes the importance of implementing strategies to address the class imbalance effectively.

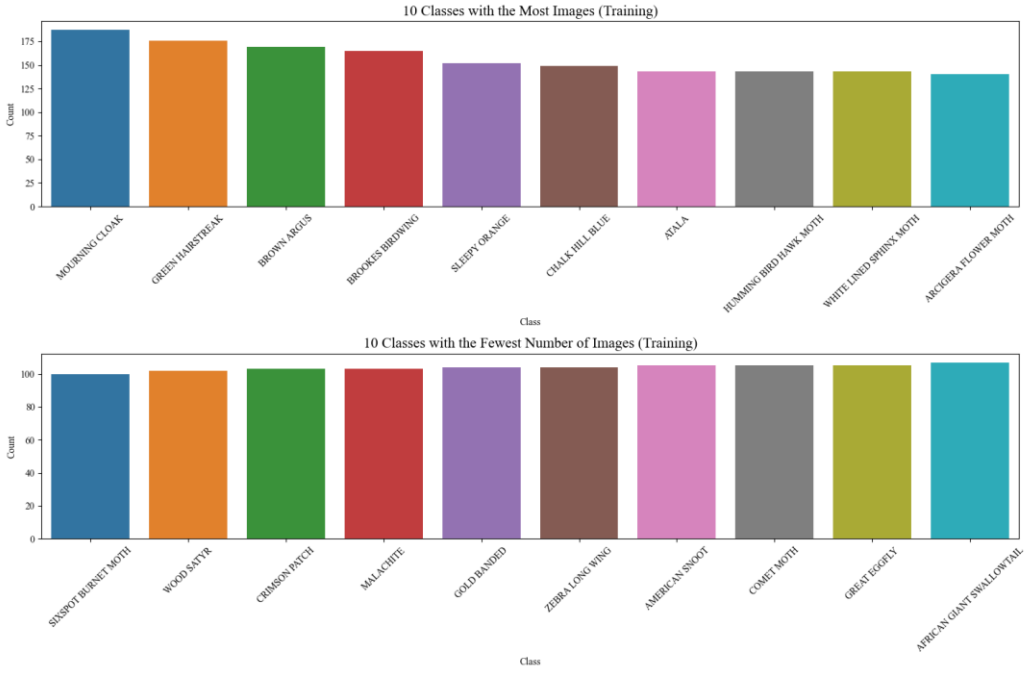

Figure 4 Top and Bottom Classes Visualization

2.3 Data Preprocessing

The data pre-processing phase combined several strategies to improve image quality and address dataset problems. Following established normalizing techniques [16], all images were rescaled to the [0,1] range by dividing pixel values by 255, resulting in more consistent input data ranges and more efficient convergence during model training. To resolve the identified class imbalance, a complete weighting technique was created based on inverse class frequency calculation [17], which prioritized underrepresented species during the learning phase. This method assigned a maximum weight of 1.0 to the least represented class *Sixspot Burnet Moth* with 100 samples, while the most abundant class *Mourning Cloak* with 187 samples received approximately 0.53, effectively balancing learning attention across all classes without requiring physical dataset modification. The implementation used TensorFlow's built-in class weighting mechanisms during model training, as

well as automated class indices generation via directory-based data loading methods, to ensure consistent mapping between butterfly species names and numerical representations throughout all processing stages. This unified method to pre-processing laid a solid foundation for future model building by ensuring data quality while maintaining class representation fidelity.

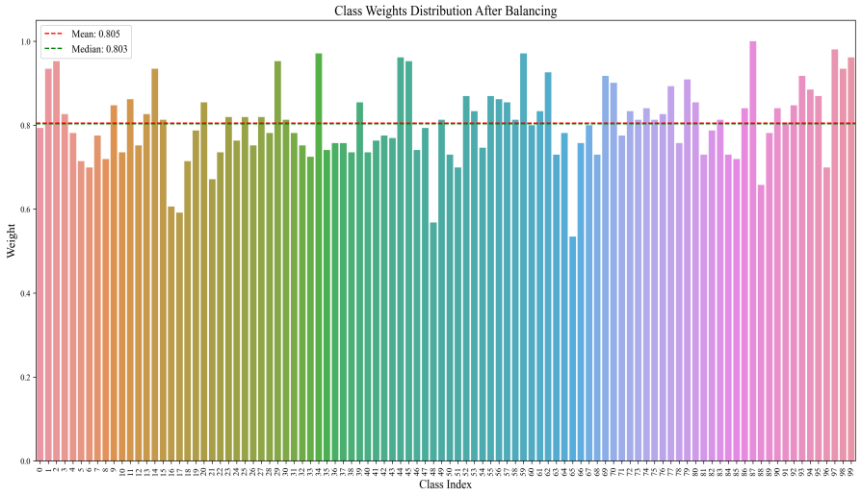

Figure 5 Class Weight Distribution After Balancing Visualization

Figure 5 illustrates the distribution of class weights after balancing, calculated using inverse class frequency to address the dataset's class imbalance. The visualization highlights the variation in class weights, which were assigned to ensure that underrepresented species received more focus during model training. The mean and median values, marked in the chart, provide additional insight into how the weight adjustment balanced the dataset across all butterfly species, laying a strong foundation for future model development.

2.4 Data Augmentation

A thorough data augmentation pipeline was created using TensorFlow's ImageDataGenerator module to improve model generalization and tackle the issue of limited training samples. In accordance with established augmentation methodologies [18], the system implemented several real-time transformations during training, including rotations of up to ±40° to account for diverse butterfly orientations, width and height adjustments of 20% to simulate different placements within frames, shear transformations of 20% to introduce perspective variations, and zoom operations within a 20% range to mimic varying capture distances. Diversity was enhanced by employing both horizontal and vertical flipping, utilizing the nearest fill mode to effectively manage border pixels generated during transformations [19]. Each augmentation parameter was intentionally selected to represent natural differences in butterfly imaging, so enhancing the diversity of the training dataset without necessitating further data collection. This method allowed the models to enhance their feature identification abilities, resulting in improved generalization to novel specimens [18]. A crucial methodological aspect was the exclusive use of augmentation techniques to the training set, while the validation and test sets were left unaltered to avert data leakage and maintain the integrity of model evaluation outcomes. The strategic application of data augmentation greatly enhanced the models'

capacity to identify butterflies in various orientations, sizes, and viewpoints, hence increasing classification performance for all 100 target species.

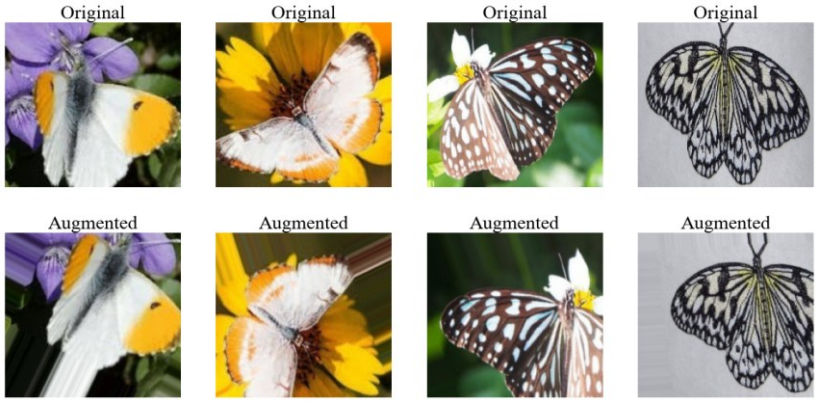

Figure 6 Comparison of Original and Augmented Images

Figure 6 illustrates the comparison between original and augmented photos, highlighting the efficacy of the data augmentation pipeline employed during training. The original photos are displayed above, while the augmented versions, generated through various transformations like rotations, scaling, flipping, and shifting, are presented below. These modifications facilitate the simulation of various butterfly orientations, sizes, and perspectives, thereby substantially augmenting the model's capacity to generalize and enhance classification performance across all 100 target species without necessitating further data gathering.

2.5 Model Architectures

This study implemented two complementary convolutional neural network architectures: MobileNetV2 and Xception, selected for their balance of computational efficiency and classification accuracy. MobileNetV2 represents a lightweight architecture specifically designed for resource-constrained environments [20], featuring inverted residual blocks with bottleneck layers and depthwise separable convolutions that decompose standard convolutions into depthwise and pointwise operations. Figure 7 shows the MobileNetV2 architecture's inverted residual blocks. On the left are stride=1 blocks with skip links, and on the right are stride=2 blocks. These blocks are the most important part of the network. They use depthwise separable convolutions to make the network much faster while keeping the richness of the features [21]. This innovative structure significantly reduces computational demands while maintaining high accuracy, with the implemented model containing approximately 2.3 million non-trainable parameters with ImageNet pre-trained weights [22]. The architecture leverages linear bottlenecks between expanded representations and skip connections that enhance gradient flow during training, making it particularly suitable for deployment scenarios where processing efficiency is critical [23].

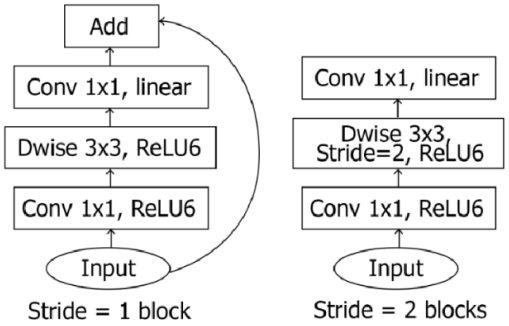

Figure 7 MobileNetV2 Architecture [21]

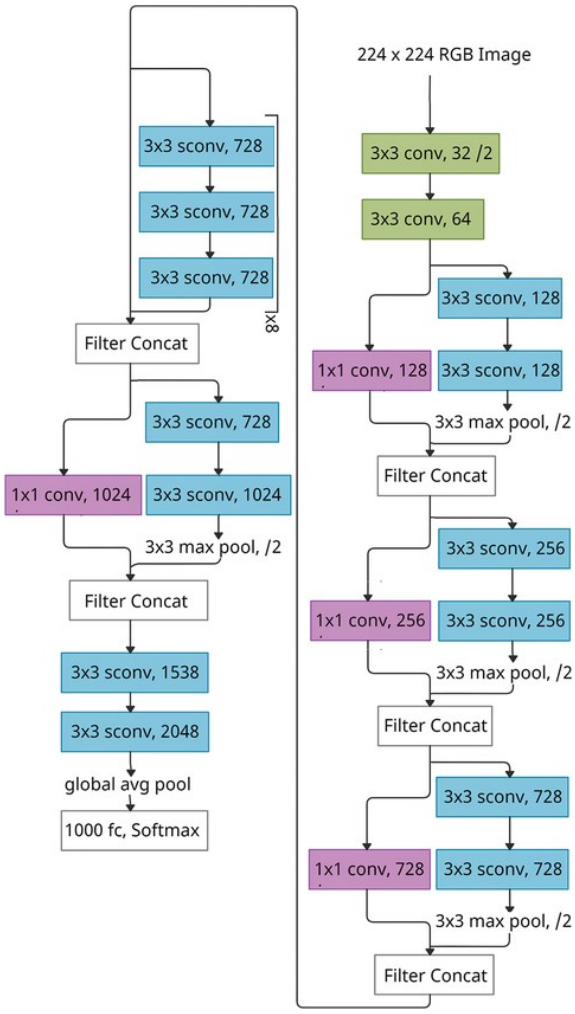

Figure 8 Xception Architecture [24]

Figure 8 depicts the entire Xception architecture used in this study, highlighting the three separate flows. The diagram depicts the entry flow (top), which includes initial convolutions and increasing filter depth, the middle flow (bottom left), which contains eight repeated blocks of depthwise separable convolutions, and the exit flow (bottom right), which transitions to higher-dimensional feature representations before classification. This layered method allows for effective cross-channel correlations while being computationally feasible. Xception provides a deeper architectural alternative that extends the concept of depthwise separable convolutions with additional

cross-channel correlations. The architecture is organized into three primary segments: entry flow with initial convolutional layers, middle flow containing repeated depthwise separable convolutions, and exit flow for transition to classification layers [25]. Featuring enhanced depthwise separable convolutions and more extensive residual connections, the Xception base model implemented in this study utilized approximately 20.8 million non-trainable parameters, reflecting its substantially deeper structure. Despite its increased complexity, Xception achieves high accuracy with relatively moderate parameter count compared to other high-performance models, outperforming several alternatives in fine-grained image classification tasks [26]. Both architectures were implemented through transfer learning, utilizing pre-trained ImageNet weights to leverage generalized visual feature extraction capabilities while enabling domain-specific adaptation through custom classification heads designed for the butterfly species identification task.

2.6 Model Development

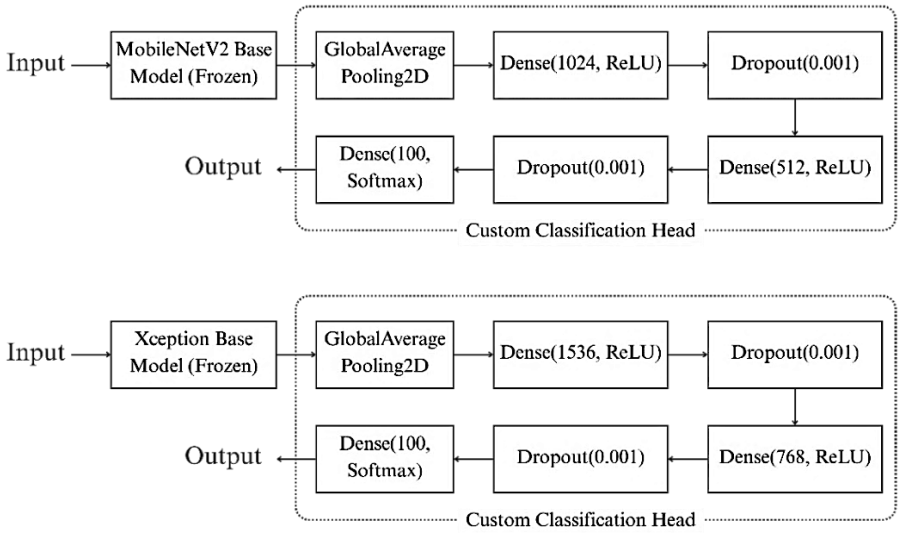

Figure 9 Modified Model Architecture

The model development process employed a systematic approach to adapt pre-trained architectures for butterfly species classification, following established practices in machine learning [27]. For both MobileNetV2 and Xception models, the base architectures were imported with pre-trained ImageNet weights, and all base layers were frozen to preserve their generalized feature extraction capabilities while allowing for domain-specific adaptation. The MobileNetV2 model was extended with a custom classification head specifically designed for butterfly classification, incorporating a global average pooling layer to reduce spatial dimensions and parameter count, followed by two fully connected layers with 1024 and 512 neurons respectively, each using ReLU activation. Lightweight dropout (0.001) was strategically applied between these layers to minimize overfitting while maintaining effective information flow. This architecture culminated in a final classification layer with softmax activation for the 100 butterfly species, with the model compiled using the Adam optimizer (initial learning rate=0.001) and categorical cross-entropy loss function. As illustrated in the upper portion of Figure 9, MobileNetV2's custom classification head maintains computational efficiency with approximately 1.89 million trainable parameters while preserving the

frozen base model's pre-trained feature extraction capabilities, creating a balanced implementation suitable for deployment in resource-constrained environments.

The lower portion of Figure 9 shows Xception's custom classification head with proportionally larger dense layers and approximately 4.4 million trainable parameters, designed to maximize the model's ability to utilize richer feature representations while maintaining the same structural pattern for systematic comparison. The Xception model received a proportionally larger custom classification head with expanded dense layers (1536 and 768 neurons) to match its increased feature extraction capacity, while maintaining the same structural pattern and dropout rate (0.001) as the MobileNetV2 implementation. This design choice accommodated the richer feature representations produced by Xception's more complex base architecture, with a lower initial learning rate (0.0005) implemented to facilitate more gradual optimization through the deeper network. Both model development approaches prioritized effective adaptation to the butterfly classification domain while preserving the core feature extraction capabilities of the pre-trained architectures. The custom classification heads were deliberately designed with sufficient capacity to capture species-specific discrimination features while avoiding excessive complexity that might lead to overfitting, resulting in MobileNetV2 having approximately 1.89 million trainable parameters (out of 4.15 million total) and Xception having approximately 4.4 million trainable parameters (out of 25.27 million total).

2.7 Model Training

The training process for the butterfly classification models employed carefully selected parameters to balance computational efficiency with model performance. A batch size of 32 and maximum 50 epochs were utilized for both MobileNetV2 and Xception architectures, though neither required the full epoch allocation due to effective early stopping mechanisms. MobileNetV2 completed training after 34 epochs, demonstrating the efficiency of the optimization strategies implemented [28]. The dataset's size of 12,594 training images across 100 butterfly species influenced these parameter choices, providing sufficient iterations for learning while preventing excessive computational overhead.

Both models benefited from comprehensive optimization techniques tailored to their specific architectures. Model checkpointing was implemented with validation accuracy monitoring and the save-best-only parameter enabled, ensuring only the highest-performing model versions were preserved during the training process. Early stopping mechanisms were configured differently between models, with MobileNetV2 using a patience parameter of 10 epochs while the more complex Xception architecture was allowed additional convergence time with 15 epochs of patience [29]. Learning rate reduction schedules were similarly customized, with MobileNetV2 employing a reduction factor of 0.2 after 5 epochs without improvement (minimum learning rate of 1e-6), while Xception utilized a more aggressive 0.1 reduction factor after 7 stagnant epochs, with a lower floor of 1e-7. These adaptations proved essential given Xception's significantly larger parameter count—approximately 20.8 million non-trainable parameters compared to MobileNetV2's 2.3 million [30].

The models were compiled with the Adam optimizer but used different initial learning rates to accommodate their architectural differences. MobileNetV2 began with a learning rate of 0.001, while Xception started lower at 0.0005 to provide more stable training for its more complex structure. Both

employed categorical cross-entropy loss functions and accuracy metrics, with class weights implemented to address the inherent imbalance in the butterfly species dataset. These configurations resulted in MobileNetV2 achieving 93.20% test accuracy, while Xception reached a marginally superior 93.40%, demonstrating that thoughtfully implemented optimization approaches can yield comparable results despite significant differences in model complexity [31].

The training process leveraged GPU acceleration through TensorFlow, confirmed through environment checks that identified a single available GPU with CUDA support. This hardware configuration enabled efficient parallel processing for both models, though Xception naturally demanded more computational resources due to its larger parameter count and more complex architecture [32]. The successful implementation of these training procedures demonstrates how carefully configured parameters and optimization techniques can produce high-performing models for specialized image classification tasks, even with moderately-sized datasets and limited hardware resources. The marginal performance difference between the models (0.20%) despite their substantial architectural differences highlights the effectiveness of the training methodology and optimization strategies employed.

2.8 Evaluation

Both CNN architectures were evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score, with targeted visualization strategies to manage the complexity of the 100-class problem [33]. Instead of examining the entire 100×100 confusion matrix, the implementation concentrated on the top 10 most represented classes and determined the 10 worst-performing classes using per-class accuracy computation [34][35]. This selective approach, combined with sample prediction visualizations with color-coded results, allowed for efficient identification of misclassification patterns between visually similar butterfly species while avoiding information overload, revealing similar classification challenges across both model architectures despite structural differences [36][37]. To quantify model performance with precision, the research utilized a suite of complementary metrics, each providing unique insights into classification behavior. The overall classification effectiveness was measured using accuracy, which represents the ratio of correctly classified instances to the total number of test samples, as defined in Equation (1).

$$Accuracy = \frac{\Sigma_{i=1}^{100} TP_i}{Total\ test\ samples} = \frac{Number\ of\ correctly\ classified\ butterfly\ species}{500} \quad (1)$$

For a more granular assessment of model behavior, precision was calculated for each butterfly species *i*. Precision quantifies the exactness of positive predictions by measuring the proportion of correctly identified instances among all predictions for a specific class, as demonstrated in Equation (2).

$$Precision_i = \frac{TP_i}{TP_i + FP_i} = \frac{Correctly\ identified\ species_i}{Total\ butterflies\ predicted\ as\ species_i} \quad (2)$$

Complementing precision, recall was employed to evaluate the model's completeness in identifying all relevant instances. Equation (3) presents the mathematical formulation of recall, which measures the proportion of actual positives that were correctly identified by the model.

$$Recall_i = \frac{TP_i}{TP_i + FN_i} = \frac{Correctly\ identified\ species_i}{Total\ actual\ butterfly\ of\ species_i} \quad (3)$$

To balance the trade-off between precision and recall, the F1-score was calculated according to Equation (4). This metric provides a harmonic mean of both precision and recall, proving particularly valuable for species with classification challenges.

$$F1\ Score_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i} \quad (4)$$

These metrics were calculated individually for all 100 butterfly species, then aggregated using both macro-averaging and weighted-averaging to provide comprehensive performance assessment across the diverse butterfly taxonomy while accommodating species-specific classification challenges.

3 Results and Discussion

The comparative analysis of MobileNetV2 and Xception architectures for butterfly species classification yielded several significant findings. Both models demonstrated robust performance, with MobileNetV2 achieving 93.20% test accuracy and Xception reaching 93.40%. This minimal difference (0.20%) despite substantial architectural disparity represents a key finding, suggesting that lightweight architectures can achieve comparable results to more complex models in specialized classification tasks. Figure 10 visualizes the accuracy comparison between both architectures, illustrating how the minimal performance difference (0.20%) occurs despite significant differences in model complexity. This visualization affirms MobileNetV2's efficiency in achieving competitive performance with a substantially simpler architecture.

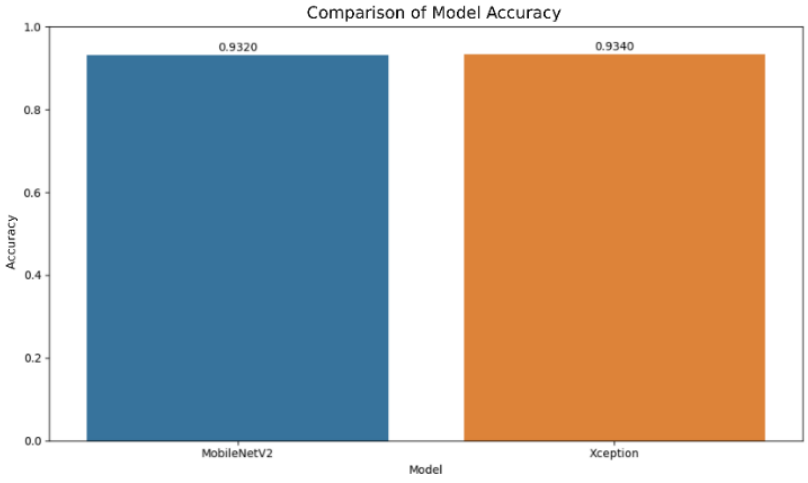

Figure 10 Model Accuracy Comparison

Both architectures exhibited consistent classification metrics across the 100 butterfly species, with identical precision (0.94), recall (0.93), and F1-score (0.93) in both macro-averaged and weighted-averaged measurements. This balanced performance validates the effectiveness of the

implemented class weighting strategy in mitigating dataset imbalance, where training samples ranged from 100 to 187 images per species.

To validate the significance of performance differences between architectures, comprehensive statistical testing was conducted using McNemar's test and paired t-test analysis. McNemar's test examined the disagreement between model predictions, revealing no statistically significant difference ($\chi^2 = 0.9730$, $p = 0.3239$, $\alpha = 0.05$). The test identified 451 cases where both models were correct, 15 cases where MobileNetV2 was correct but Xception was wrong, 22 cases where Xception was correct but MobileNetV2 was wrong, and 12 cases where both models failed. Additionally, paired t-test analysis on per-class accuracy showed no significant difference between architectures ($t = -1.1224$, $p = 0.2644$). These statistical results confirm that despite Xception's marginally higher accuracy (93.40% vs. 93.20%), the performance difference is not statistically significant, supporting the practical equivalence of both architectures for butterfly species classification tasks.

Comprehensive computational benchmarking revealed significant efficiency advantages for MobileNetV2 across multiple metrics. Inference time analysis showed MobileNetV2 achieving 47.78 ± 5.64 ms per image compared to Xception's 51.22 ± 8.86 ms, representing a 1.1× speed advantage with lower variance. Model size comparison demonstrated dramatic differences, with MobileNetV2 requiring only 15.82 MB compared to Xception's 96.38 MB (6.1× reduction). Parameter analysis revealed MobileNetV2's efficiency with 4.15 million total parameters (1.89 million trainable) versus Xception's 25.27 million total parameters (4.40 million trainable), representing a 6.1× reduction in model complexity. Batch processing benchmarks showed similar efficiency patterns, with MobileNetV2 processing multiple images 1.9× faster than Xception. Memory usage analysis indicated minimal differences (0.42 MB vs. 0.49 MB), suggesting both architectures are suitable for resource-constrained deployments, though MobileNetV2 maintains superior overall efficiency characteristics.

Table 1 Comprehensive Performance and Efficiency Comparison

Metrics	MobileNetV2	Xception	Ratio (Xception/MobileNetV2)
Test Accuracy	0.9320	0.9340	1.015
Model Size (MB)	15.82	96.38	6.1x
Total Param (M)	4.15	25.27	6.1x
Trainable Param (M)	1.89	4.40	2.3x
Avg Inference Time (ms)	47.78 ± 5.64	51.22 ± 8.86	1.1x
Batch Inference Time (ms)	102.21	202.49	1.9x
Perfect Classification Species	48	52	1.1x

Figure 11 and 12 illustrates the training curves of both models, visualizing the dynamics of accuracy and loss throughout the optimization process. The graph demonstrates how MobileNetV2 achieved convergence more rapidly at epoch 34, while Xception required 50 epochs with a more gradual pattern of performance improvement.

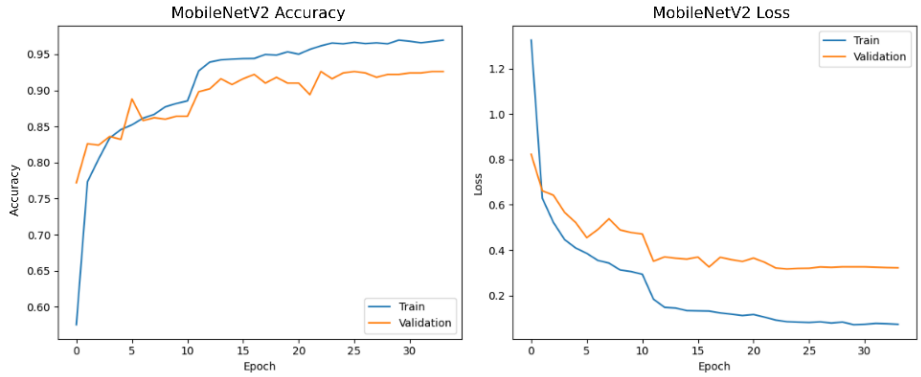

Figure 11 MobileNetV2 Training History

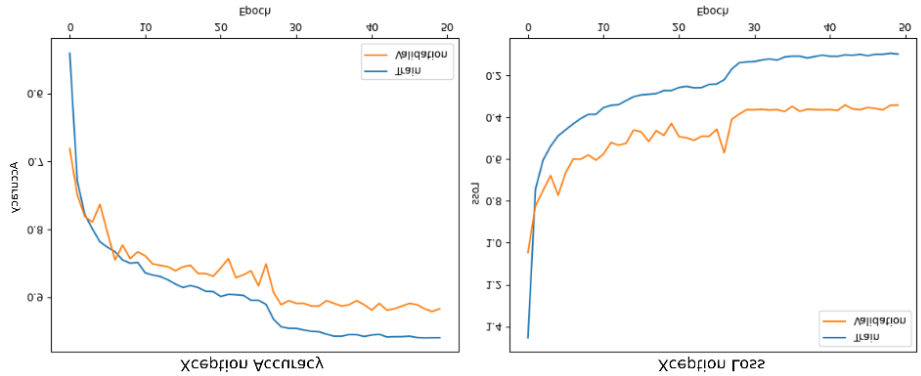

Figure 12 Xception Training History

The validation loss dynamics observed in Figure 12 reveal important insights into the transfer learning optimization behavior of the Xception architecture. The notable fluctuations around epochs 30-35, particularly the temporary loss spike, reflect the complex interaction between pre-trained ImageNet features and domain-specific butterfly classification adaptation. This phenomenon coincides with the learning rate reduction schedule implementation, where the aggressive reduction factor (0.1) temporarily disrupts the optimization trajectory before enabling more refined parameter adjustment. Such oscillatory behavior is characteristic of deeper architectures with extensive parameter spaces, which require more careful optimization compared to lightweight models. The subsequent stabilization and continued improvement demonstrate successful domain adaptation, with the model ultimately achieving superior test performance despite these temporary optimization challenges. This pattern validates the importance of extended patience parameters (15 epochs) in early stopping criteria for complex transfer learning scenarios.

Species-specific analysis revealed performance patterns aligned with taxonomic visual characteristics. Perfect classification (100% accuracy) was achieved for species with distinctive visual features Atlas Moth, Monarch, and Red Admiral, while both models struggled with visually similar species, particularly among blue butterflies such as Chalk Hill Blue and Adonis as well as those with similar wing patterns. This pattern aligns with theoretical expectations in fine-grained classification, where subtle distinguishing features become critical determinants of model performance. Figure 13,14, and 15 presents a comparison of the highest and lowest performing

species for both models. This visualization reveals how specific morphological characteristics correlate with classification accuracy, with species having distinctive patterns such as Monarch and Atlas Moth consistently achieving perfect accuracy, while species with more common patterns or higher intra-species variation present greater classification challenges.

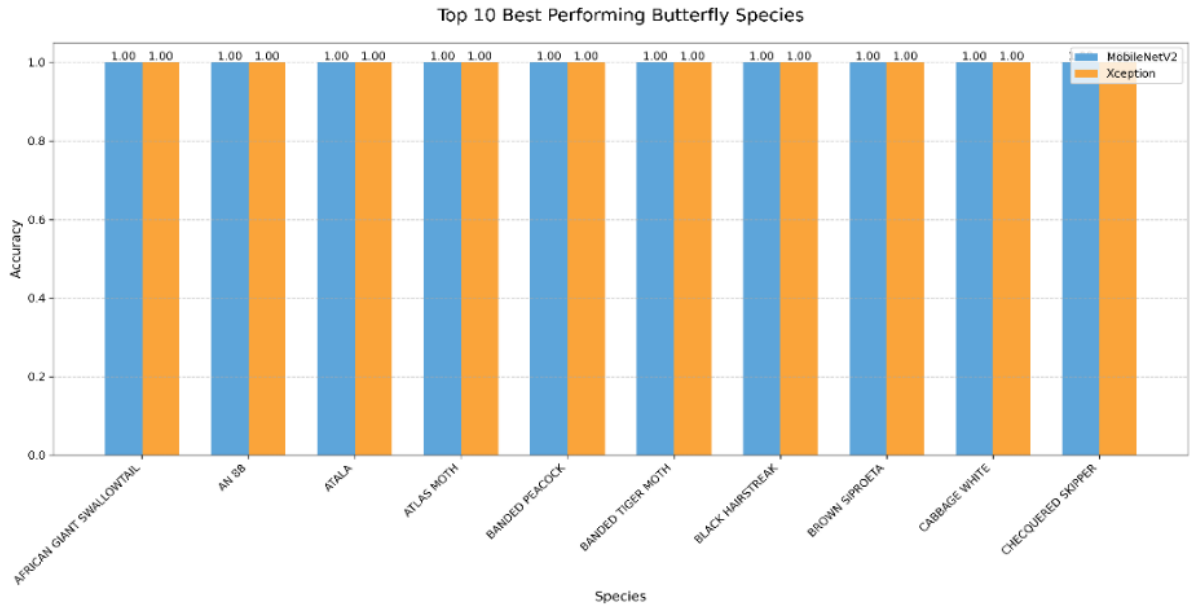

Figure 13 Best Performing Species

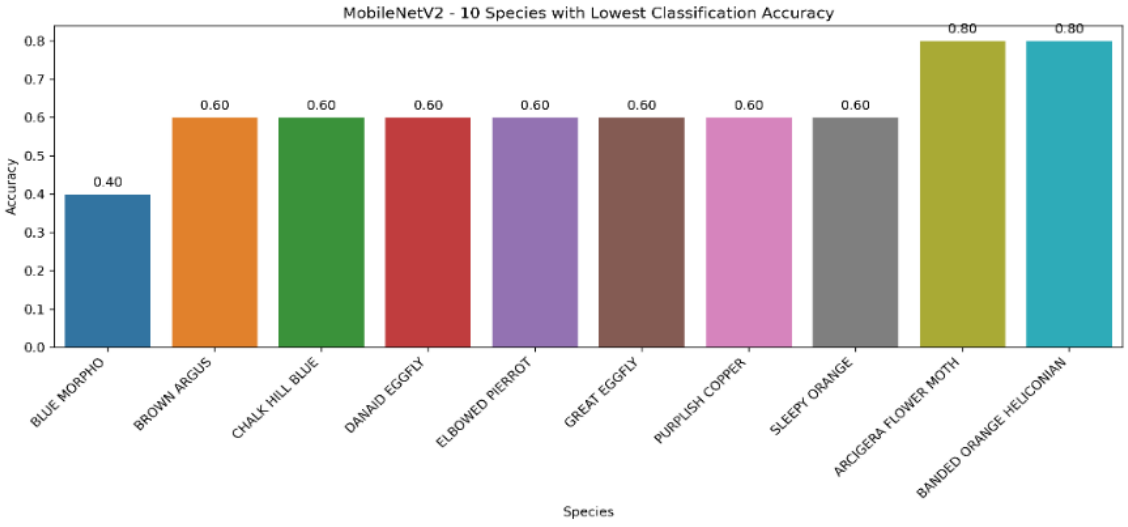

Figure 14 Worst Performing Species (MobileNetV2)

Figure 16 and Figure 17 presents confusion matrixes for both models, revealing the primary misclassification patterns among species. The dominant diagonal pattern confirms overall high accuracy, while the off-diagonal areas identify pairs of species frequently confused with each other, particularly among butterfly families with similar visual characteristics. Due to better understand the visual context of classification successes and failures, Figure 18 presents sample predictions with actual and predicted labels. These examples demonstrate cases where distinctive visual features

resulted in accurate classification (marked in green), as well as cases where similarities in wing patterns or posture led to misclassification (marked in red).

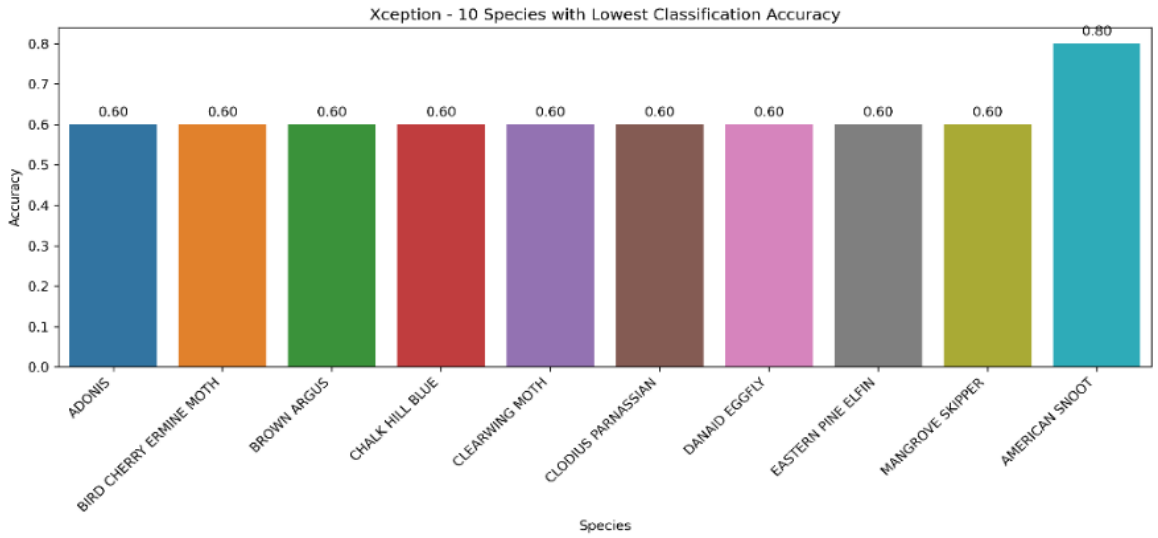

Figure 15 Worst Performing Species (Xceptio

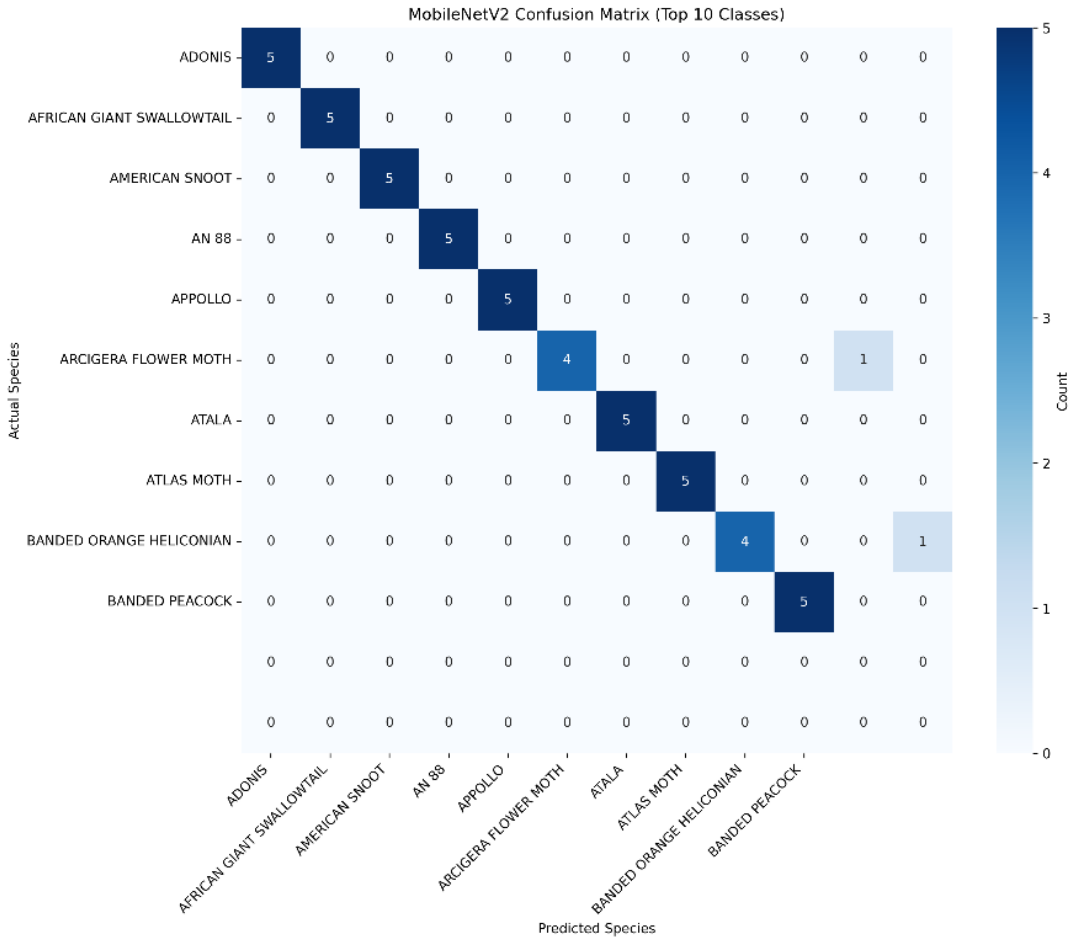

Figure 16 MobileNetV2 Confusion Matrix

These findings directly address the study's primary objective of comparing MobileNetV2 and Xception architectures for butterfly classification. The results demonstrate that transfer learning with custom classification layers enables both architectures to achieve excellent performance, with MobileNetV2 offering substantial efficiency advantages without significant accuracy sacrifice. Such findings have important implications for practical implementation of butterfly identification systems in resource-constrained environments. The observed performance compares favorably with previous research, positioning in the upper tier of reported accuracy ranges for butterfly classification tasks.

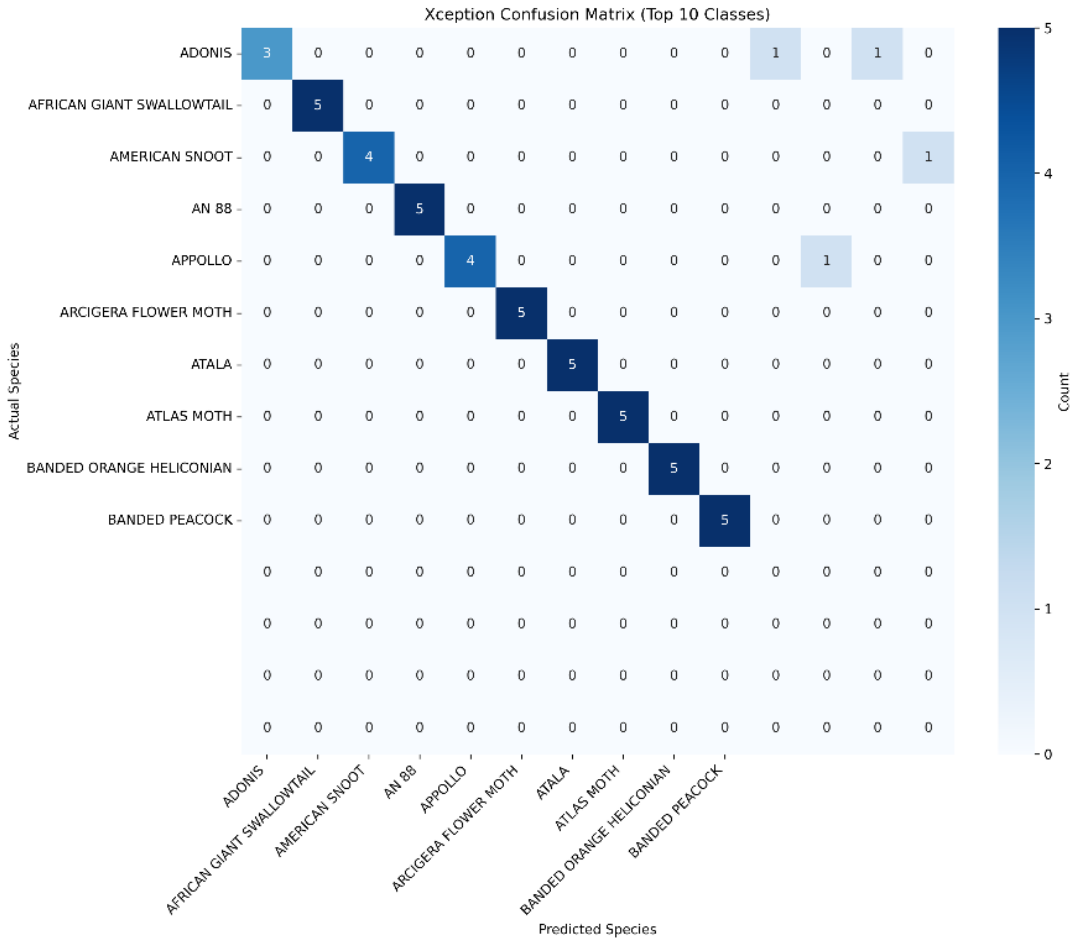

Figure 17 Xception Confusion Matrix

This study results show MobileNetV2 achieved 93.20% accuracy which is slightly lower than Kaur's study in 2024 which reported 94.6% accuracy using MobileNetV2 for 75 butterfly species. However, several factors may explain this difference. First, our study classified a larger taxonomy with 100 species compared to 75 species, increasing classification complexity. Second, our implementation focused on architectural comparison rather than single-model optimization. Third, our dataset distribution with 12,594 training images across 100 classes differs from Kaur's approach with 9,285 training images across 75 classes. Notably, while Kaur's study emphasized mobile deployment optimization, our research provides additional insights through architectural comparison, demonstrating that with appropriate optimization, lightweight architectures can approach the

performance of more complex models for specialized visual domains, even with expanded taxonomic scope [5]. The effective implementation of class weighting and data augmentation strategies significantly contributed to model performance, supporting established research on addressing imbalanced datasets in biological classification tasks. The approach achieved balanced performance across all species despite training sample variations, demonstrating practical application of theoretical balancing techniques in entomological classification.

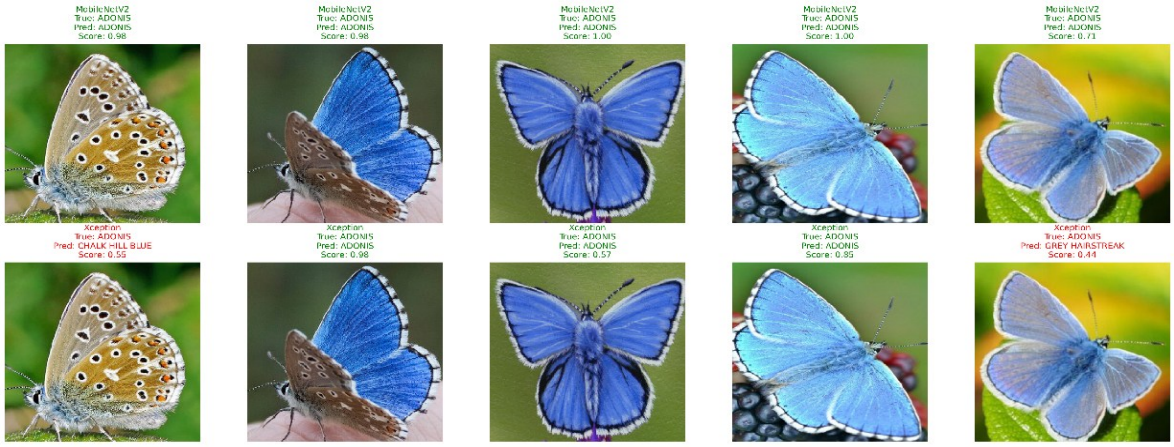

Figure 18 Sample Predictions

The significance of these findings extends to multiple domains. For conservation applications, the high accuracy across 100 species provides valuable tools for monitoring butterfly populations, particularly for rare or endangered species. The efficiency of MobileNetV2 makes it suitable for deployment in mobile applications, enabling citizen science initiatives through automated identification assistance. The successful classification across a comprehensive butterfly taxonomy supports detailed ecological studies requiring species-level identification precision. In conclusion, this study demonstrates that both MobileNetV2 and Xception architectures achieve excellent butterfly species classification performance when appropriately optimized, with MobileNetV2 offering compelling advantages in computational efficiency with negligible accuracy sacrifice. These findings support implementation decisions for automated butterfly identification systems in practical conservation and research applications, highlighting the effectiveness of transfer learning and custom classification layers for specialized visual recognition tasks.

4 Conclusion

This comparative analysis of lightweight and complicated CNN architectures for butterfly species categorization illustrates the effectiveness of transfer learning methods for specific taxonomic classification problems. The study effectively achieved the main goal by demonstrating that architectural complexity does not inherently connect to classification performance in domain-specific image recognition tasks. The results offer compelling scientific rationale for utilizing lightweight architectures in specialized visual classification tasks, which is particularly important for practical entomological identification systems with constrained computational resources. The minimal

performance variation between designs, despite significant parameter differences, questions traditional beliefs regarding the complexity needs of models for fine-grained classification tasks. This study methodologically contributes to the field by demonstrating the effective implementation of class weighting algorithms that provide balanced performance among taxonomically different species. The persistent trend of successful classification with visually dissimilar species and difficulties with visually similar ones highlights the essential connection between visual taxonomic traits and machine learning classification efficacy.

Future research should investigate ensemble methods that utilize the complementing advantages of various architectural approaches, as indicated by the distinct error patterns identified among models. Additional research on attention mechanisms tailored to emphasize taxonomically significant butterfly characteristics may enhance classification precision for visually like species. Furthermore, the creation of region-specific models concentrating on butterflies from distinct geographical locales signifies another viable avenue for targeted conservation efforts. The wider implications reach beyond entomology to conservation biology, citizen science projects, and computer vision techniques, illustrating how optimized lightweight architectures can democratize access to sophisticated classification tools in resource-limited research and conservation settings.

Bibliography

[1] M. A. H. Saedan, M. Kassim, and A. F. Abd Aziz, "Biological Butterfly Characterization with Mobile System Using Convolutional Neural Network (CNN) Classify Image," *International Journal of Interactive Mobile Technologies*, vol. 18, no. 7, pp. 125–138, Apr. 2024, doi: 10.3991/ijim.v18i07.46267.

[2] F. Rajeena P. P. *et al.*, "A Novel Method for the Classification of Butterfly Species Using Pre-Trained CNN Models," *Electronics (Basel)*, vol. 11, no. 13, p. 2016, Jun. 2022, doi: 10.3390/electronics11132016.

[3] D. S. Singh, A. Kumar Pandey, A. S. Singh, and A. K. Mishra, "Butterfly Species Recognition Using Convolutional Neural Network," *IRE Journals*, vol. 7, no. 6, Dec. 2023.

[4] M. Santhiya, S. Karpagavalli, "Integration of a Dual Hybrid Deep Convolutional Neural Network Framework for Insect Taxonomic Classification," *Communications on Applied Nonlinear Analysis*, vol. 32, no. 9s, pp. 2244–2257, Mar. 2025, doi: 10.52783/cana.v32.4510.

[5] A. Kaur, "Nature Meets Technology: Revolutionizing Butterfly Classification with MobileNetV2," in *2024 9th International Conference on Communication and Electronics Systems (ICCES)*, IEEE, Dec. 2024, pp. 1449–1454. doi: 10.1109/ICCES63552.2024.10859734.

[6] H. T. Adityawan, O. Farroq, S. Santosa, H. M. M. Islam, M. K. Sarker, and D. R. I. M. Setiadi, "Butterflies Recognition using Enhanced Transfer Learning and Data Augmentation," *Journal of Computing Theories and Applications*, vol. 1, no. 2, pp. 115–128, Nov. 2023, doi: 10.33633/jcta.v1i2.9443.

[7] K. Soman, C. Kokate, A. Mohite, A. Vispute, O. More, and Prof. Z. K. Mundargi, "Statistical Tests for Comparing Machine Learning Algorithms," *Int J Res Appl Sci Eng Technol*, vol. 10, no. 12, pp. 628–633, Dec. 2022, doi: 10.22214/ijraset.2022.47955.

[8] R.-Y. Sun, "Optimization for Deep Learning: An Overview," *Journal of the Operations Research Society of China*, vol. 8, no. 2, pp. 249–294, Jun. 2020, doi: 10.1007/s40305-020-00309-6.

[9] F. Zhuang *et al.*, "A Comprehensive Survey on Transfer Learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.

[10] Z. Lin, J. Jia, W. Gao, and F. Huang, "Fine-grained visual categorization of butterfly specimens at sub-species level via a convolutional neural network with skip-connections," *Neurocomputing*, vol. 384, pp. 295–313, Apr. 2020, doi: 10.1016/j.neucom.2019.11.033.

[11] T. Xi, J. Wang, Y. Han, C. Lin, and L. Ji, "Multiple butterfly recognition based on deep residual learning and image analysis," *Entomol Res*, vol. 52, no. 1, pp. 44–53, Jan. 2022, doi: 10.1111/1748-5967.12564.

[12] D. Elreedy, A. F. Atiya, and F. Kamalov, "A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning," *Mach Learn*, vol. 113, no. 7, pp. 4903–4923, Jul. 2024, doi: 10.1007/s10994-022-06296-4.

[13] P. Soltanzadeh, M. R. Feizi-Derakhshi, and M. Hashemzadeh, "Addressing the class-imbalance and class-overlap problems by a metaheuristic-based under-sampling approach," *Pattern Recognit*, vol. 143, p. 109721, Nov. 2023, doi: 10.1016/j.patcog.2023.109721.

[14] L. Li, H. He, and J. Li, "Entropy-based Sampling Approaches for Multi-Class Imbalanced Problems," *IEEE Trans Knowl Data Eng*, vol. 32, no. 11, pp. 2159–2170, Nov. 2020, doi: 10.1109/TKDE.2019.2913859.

[15] Z. Sun, W. Ying, W. Zhang, and S. Gong, "Undersampling method based on minority class density for imbalanced data," *Expert Syst Appl*, vol. 249, p. 123328, Sep. 2024, doi: 10.1016/j.eswa.2024.123328.

[16] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Appl Soft Comput*, vol. 97, p. 105524, Dec. 2020, doi: 10.1016/j.asoc.2019.105524.

[17] T. Huynh, A. Nibali, and Z. He, "Semi-supervised learning for medical image classification using imbalanced training data," *Comput Methods Programs Biomed*, vol. 216, p. 106628, Apr. 2022, doi: 10.1016/j.cmpb.2022.106628.

[18] L. F. Sánchez-Peralta, A. Picón, F. M. Sánchez-Margallo, and J. B. Pagador, "Unravelling the effect of data augmentation transformations in polyp segmentation," *Int J Comput Assist Radiol Surg*, vol. 15, no. 12, pp. 1975–1988, Dec. 2020, doi: 10.1007/s11548-020-02262-4.

[19] L. Nanni, M. Paci, S. Brahnam, and A. Lumini, "Feature transforms for image data augmentation," *Neural Comput Appl*, vol. 34, no. 24, pp. 22345–22356, Dec. 2022, doi: 10.1007/s00521-022-07645-z.

[20] M. C. Bingol and G. Bilgin, "Prediction of Chicken Diseases by Transfer Learning Method," *International Scientific and Vocational Studies Journal*, vol. 7, no. 2, pp. 170–175, Dec. 2023, doi: 10.47897/bilmes.1396890.

[21] S. G. Brucal *et al.*, "Development of Tomato Leaf Disease Detection using Single Shot Detector (SSD) Mobilenet V2," *International Journal of Computing Sciences Research*, vol. 7, pp. 1857–1869, Jan. 2023, doi: 10.25147/ijcsr.2017.001.1.136.

[22] S. K. T. S, P. A, A. P. K, and S. Alagammal, "A Comparative Study on Plant Classification Performance using Deep Learning Optimizers," in *2021 Emerging Trends in Industry 4.0 (ETI 4.0)*, IEEE, May 2021, pp. 1–9. doi: 10.1109/ETI4.051663.2021.9619238.

[23] Y. Gulzar, "Fruit Image Classification Model Based on MobileNetV2 with Deep Transfer Learning Technique," *Sustainability*, vol. 15, no. 3, p. 1906, Jan. 2023, doi: 10.3390/su15031906.

[24] K. Srinivasan *et al.*, "Performance Comparison of Deep CNN Models for Detecting Driver's Distraction," *Computers, Materials & Continua*, vol. 68, no. 3, pp. 4109–4124, 2021, doi: 10.32604/cmc.2021.016736.

[25] A. A. Mukhlif, B. Al-Khateeb, and M. A. Mohammed, "Incorporating a Novel Dual Transfer Learning Approach for Medical Images," *Sensors*, vol. 23, no. 2, p. 570, Jan. 2023, doi: 10.3390/s23020570.

[26] P. Sobti, A. Nayyar, Niharika, and P. Nagrath, "EnsemV3X: a novel ensembled deep learning architecture for multi-label scene classification," *PeerJ Comput Sci*, vol. 7, p. e557, May 2021, doi: 10.7717/peerj-cs.557.

[27] S. Boeschoten, C. Catal, B. Tekinerdogan, A. Lommen, and M. Blokland, "The automation of the development of classification models and improvement of model quality using feature engineering techniques," *Expert Syst Appl*, vol. 213, p. 118912, Mar. 2023, doi: 10.1016/j.eswa.2022.118912.

[28] L. Shen, Y. Sun, Z. Yu, L. Ding, X. Tian, and D. Tao, "On Efficient Training of Large-Scale Deep Learning Models," *ACM Comput Surv*, vol. 57, no. 3, pp. 1–36, Mar. 2025, doi: 10.1145/3700439.

[29] K. Karthick, "Comprehensive Overview of Optimization Techniques in Machine Learning Training," *Control Systems and Optimization Letters*, vol. 2, no. 1, pp. 23–27, Feb. 2024, doi: 10.59247/csol.v2i1.69.

[30] R. Li, D. Fu, C. Shi, Z. Huang, and G. Lu, "Efficient LLMs Training and Inference: An Introduction," *IEEE Access*, vol. 13, pp. 32944–32970, 2025, doi: 10.1109/ACCESS.2024.3501358.

[31] A. Ramkumar, "Accelerating Foundational Model Training: A Systematic Review of Hardware, Algorithmic, and Distributed Computing Optimizations," *International Journal For Multidisciplinary Research*, vol. 6, no. 6, Dec. 2024, doi: 10.36948/ijfmr.2024.v06i06.32140.

[32] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, New York, NY, USA: ACM, Aug. 2020, pp. 3505–3506. doi: 10.1145/3394486.3406703.

[33] G. Canbek, T. Taskaya Temizel, and S. Sagiroglu, "BenchMetrics: a systematic benchmarking method for binary classification performance metrics," *Neural Comput Appl*, vol. 33, no. 21, pp. 14623–14650, Nov. 2021, doi: 10.1007/s00521-021-06103-6.

[34] A. Luque, M. Mazzoleni, A. Carrasco, and A. Ferramosca, "Visualizing Classification Results: Confusion Star and Confusion Gear," *IEEE Access*, vol. 10, pp. 1659–1677, 2022, doi: 10.1109/ACCESS.2021.3137630.

[35] L.-E. Pommé, R. Bourqui, R. Giot, and D. Auber, "Relative Confusion Matrix: Efficient Comparison of Decision Models," in *2022 26th International Conference Information Visualisation (IV)*, Jan. 2023. doi: 10.1109/IV56949.2022.00025.

[36] I. Markoulidakis and G. Markoulidakis, "Probabilistic Confusion Matrix: A Novel Method for Machine Learning Algorithm Generalized Performance Analysis," *Technologies (Basel)*, vol. 12, no. 7, Jul. 2024, doi: 10.3390/technologies12070113.

[37] A. Hinterreiter *et al.*, "ConfusionFlow: A model-agnostic visualization for temporal analysis of classifier confusion," *IEEE Trans Vis Comput Graph*, vol. 28, no. 2, pp. 1222–1236, Oct. 2019, doi: 10.1109/TVCG.2020.3012063.