

Evaluating the Role of PCA Dimensionality Reduction in Enhancing CNN-Based Butterfly Image Classification

Inaayah Nazhifah^{1*}, Fikri Ahmad Suganda², Arif Rizasanistri Wilotikto³,
Raihan Darwisy R⁴, M. Tasliyatul Kurbi⁵

^{1,2}Department of Information System, Faculty of Science and Technology,
Universitas Islam Negeri Sultan Syarif Kasim Riau, Indonesia

³Department of Business Information System, Faculty of Computing and Technology,
Asia Pacific University, Malaysia

⁴Department of Tafsir and Qur'anic Studies, Faculty of Ushuluddin, Al-Azhar University, Egypt

⁵Department of Business Management, Bursa Uludağ University, Turkey

E-Mail: ¹12350323132@students.uin-suska.ac.id, ²12350313386@students.uin-suska.ac.id,
³arifrizasan@gmail.com, ⁴raihandarwisy798@gmail.com, ⁵kurbiyat21@gmail.com

Received Aug 4th 2026; Revised Sep 15th 2026; Accepted Sep 28th 2026; Available Online Oct 05th 2026

Corresponding Author: Inaayah Nazhifah

Copyright © 2026 by Authors, Published by Institut Riset dan Publikasi Indonesia (IRPI)

Abstract

Butterfly species classification from digital images is a challenging computer vision task due to high intra-class variability across 75 species. Although Convolutional Neural Networks (CNN) have demonstrated strong performance in biological image classification, high-dimensional image data raises concerns about computational cost and the risk of overfitting. The main novelty of this study is a rigorous empirical evaluation of Principal Component Analysis (PCA) as a dimensionality-reduction preprocessing step in a CNN-based butterfly classification pipeline, a combination that has not been explicitly investigated in this domain. Three comparative experimental scenarios were designed: Scenario A used VGG16 transfer learning without PCA as a baseline; Scenario B applied PCA retaining 95% explained variance before a Multilayer Perceptron (MLP) classifier; and Scenario C applied PCA retaining 99% explained variance with the same MLP architecture. The performance evaluation covered accuracy, precision, recall, F1 Score, training time, inference time, and model size. Results show that Scenario A achieved the highest accuracy of 62.05% (F1-score: 0.6158), while Scenarios B and C reached 35.90% and 36.62% respectively, with training time reduced by up to 98.5% compared to the baseline. These findings reveal a clear accuracy-efficiency trade-off: PCA substantially reduces computational cost but at the expense of classification accuracy when applied as preprocessing before MLP rather than within CNN layers. This study offers practical empirical evidence to guide practitioners in selecting between accuracy-optimized and efficiency-optimized strategies when computational resources are constrained.

Keywords: Butterfly Image Classification, Convolutional Neural Network, Dimensionality Reduction, PCA

1. INTRODUCTION

More than 18,000 butterfly species have been identified across diverse habitats worldwide, making butterflies one of the most biologically diverse groups of insects. Their diversity and distribution have attracted considerable attention in fields such as biology, conservation, and environmental science, as butterfly populations are often regarded as indicators of ecosystem health [1]. Due to their ecological roles as natural pollinators and bioindicators of climate change, accurate butterfly species identification is essential [2]. Consequently, the development of fast and reliable identification systems has become increasingly important to support global biodiversity conservation efforts.

Traditionally, butterfly species identification has relied heavily on human taxonomic expertise, a process that is both time-consuming and susceptible to subjective errors. Manual classification requires substantial effort and resources and becomes particularly challenging when distinguishing species with highly similar visual characteristics or when dealing with large-scale datasets [3]. Furthermore, conventional lepidopterological studies often require researchers to capture butterflies and perform dissections to identify species and family characteristics, which may damage valuable specimens [4]. These limitations have highlighted the need for automated image-based classification systems [5].

Over the past decade, rapid advances in deep learning have created new opportunities for biological image classification. Automated image classification systems have become a major research focus, with Convolutional Neural Networks (CNN) emerging as one of the most powerful approaches [6]. CNNs learn hierarchical visual representations, from low-level features such as edges and textures to high-level features such as wing patterns and species-specific color characteristics. Alzubaidi et al. (2021) provided a comprehensive review of CNN architectures, challenges, and applications across multiple domains, confirming CNNs as the state-of-the-art approach for image recognition [7]. He et al. (2016) introduced deep residual learning, demonstrating that residual connections enable training of very deep networks with significantly improved accuracy [8]. In the context of transfer learning, models pre-trained on large-scale datasets such as ImageNet can be adapted to domain-specific tasks with limited data, making them particularly suitable for fine-grained biological image classification [9].

Several previous studies have applied CNNs to butterfly species classification with promising results. Rajeeva et al. (2022) developed a transfer learning-based CNN framework using architectures such as VGG16, VGG19, MobileNet, and ResNet50 on a dataset of 75 butterfly species, demonstrating that data augmentation consistently improved classification performance across all evaluated models [2]. Mattins et al. (2024) proposed a customized Efficient CNN (Effi-CNN) and benchmarked it against five widely used transfer learning models for automated butterfly detection and classification [3]. Teke and Elsamoly (2025) enhanced VGG16 for Turkish butterfly species identification and reported notable accuracy improvements [4]. Maftukhah et al. (2024) applied AlexNet to a multi-species butterfly dataset and demonstrated competitive accuracy through appropriate architectural tuning [10]. Xu et al. (2022) addressed fine-grained butterfly recognition using a peer learning network with distribution-aware penalty mechanisms, achieving robust discrimination despite high visual similarity among species [11]. Adityawan et al. (2023) further demonstrated that enhanced transfer learning with data augmentation strategies can substantially improve butterfly recognition performance in resource-constrained settings [12].

Despite their effectiveness, CNN models often face challenges when processing high-dimensional image data. For example, a 128×128 RGB image contains 49,152 input dimensions. High-dimensional datasets can negatively affect machine learning performance by increasing computational complexity, reducing efficiency, and raising the risk of overfitting [13]. As a result, dimensionality reduction techniques have become an important preprocessing step in machine learning and deep learning pipelines. Principal Component Analysis (PCA) is among the most widely adopted dimensionality reduction methods. By performing eigenvector decomposition on the covariance matrix, PCA transforms the original data into a new feature space while retaining only the principal components that explain the largest proportion of variance [14].

To date, no study has explicitly compared the effect of PCA as a standalone preprocessing dimensionality reduction stage on CNN performance specifically in the domain of butterfly image classification. While PCA has been applied in medical imaging [5] and hyperspectral classification [15], and CNN-based butterfly classification has been widely studied [2], [3], [4], [10], the intersection of these two methodological streams remains unexplored. This research gap constitutes the primary motivation for the present study. The main contribution of this work is an empirical, quantitative analysis of the accuracy-efficiency trade-off induced by PCA preprocessing across three controlled experimental scenarios, using a standardized 75-species butterfly dataset. The results are intended to provide actionable guidance for practitioners developing classification systems under constrained computational resources, specifically in choosing between CNN-based feature extraction and PCA-based dimensionality reduction pipelines.

2. MATERIAL AND METHOD

This study employed a comparative experimental approach to investigate the impact of applying Principal Component Analysis (PCA) as a dimensionality reduction technique during the preprocessing stage on the performance of a Convolutional Neural Network (CNN) for butterfly image classification [7]. The research methodology consisted of five main stages: dataset collection and partitioning, image preprocessing, PCA implementation in selected experimental scenarios, model training across three different experimental setups, and performance evaluation based on classification accuracy and computational efficiency. All experiments were conducted on the Google Colab platform with NVIDIA T4 GPU acceleration, using TensorFlow 2.x/Keras and Scikit-learn for model development, training, and evaluation. The research workflow adopted in this study is presented in Figure 1.

2.1. Dataset

This study utilized the Butterfly Image Classification dataset publicly available on Kaggle and curated by phuchtaiv02 (2023) [8]. The dataset contains 9,285 labeled images in JPG format representing 75 butterfly species. All images are stored in a single flat directory rather than being organized into class-specific subfolders. Label information is provided through the `Training_set.csv` file, which contains two primary attributes: image name and class label [9]. Since the dataset was originally released as part of a competition, validation and testing subsets were manually generated from the labeled training data. The dataset was

partitioned into training, validation, and test sets in a 70%, 15%, and 15% split, respectively. To maintain a balanced class distribution across all subsets, a stratified splitting strategy was applied, as shown in Figure 2.

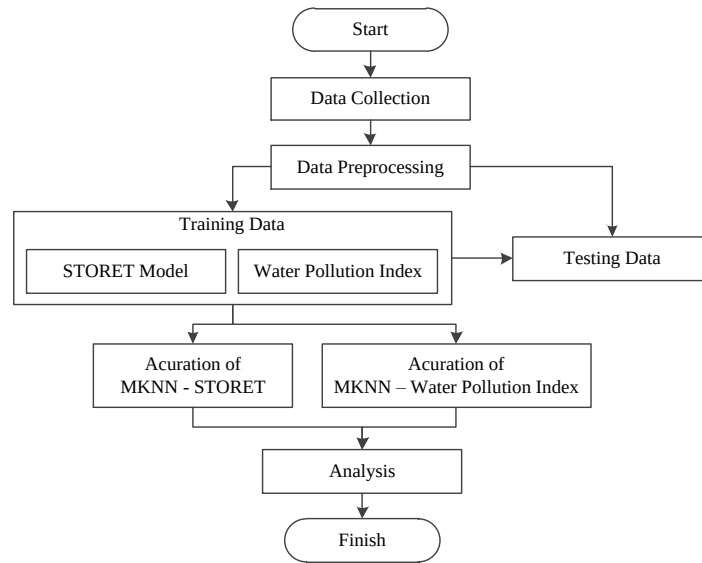


Figure 1. Research Methodology

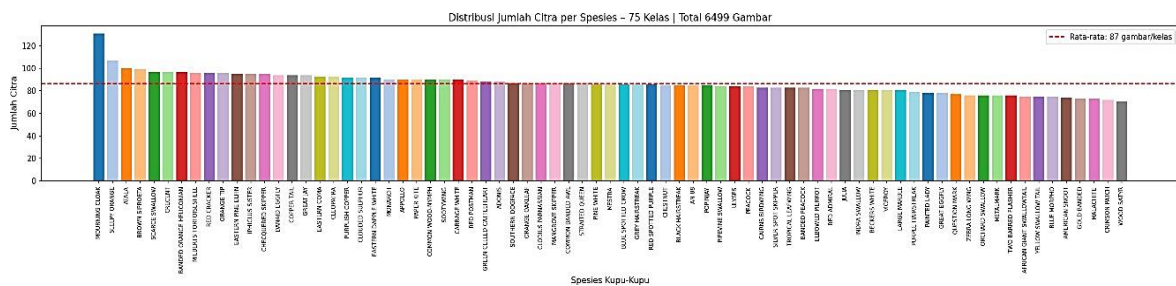


Figure 2. Class Distribution

2.2. Image Preprocessing

All images underwent a standard preprocessing procedure before being incorporated into the experimental pipeline [14]. Each image was resized to a resolution of 128×128 pixels to ensure a uniform input format and reduce memory consumption on the Google Colab platform. Pixel values were normalized from the range of $[0, 255]$ to $[0, 1]$ by dividing each value by 255, which helps accelerate model convergence during training. For Scenarios B and C, additional preprocessing steps were applied. Specifically, each image was flattened into a one-dimensional vector containing 49,152 features ($128 \times 128 \times 3$), followed by feature standardization using StandardScaler to achieve zero mean and unit variance prior to PCA transformation [10].

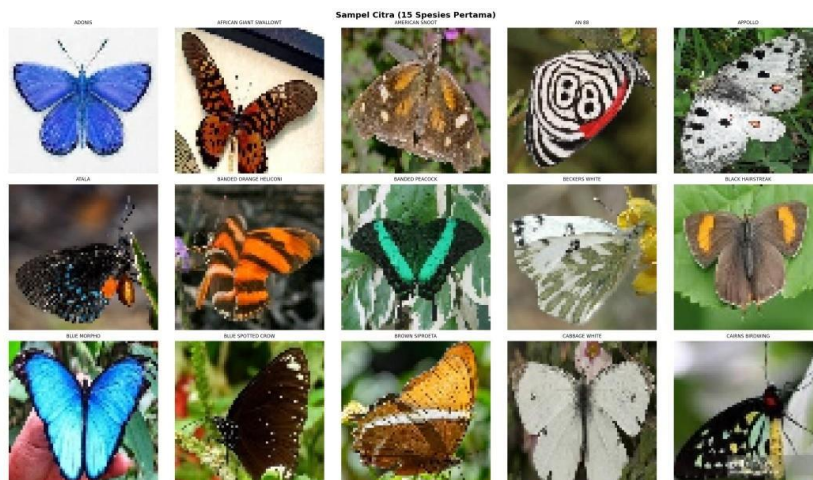


Figure 3. Images Sample

2.3. Principal Component Analysis (PCA)

PCA was employed as a preprocessing stage prior to classification in Scenarios B and C. The technique computes the covariance matrix of the standardized training data and performs eigenvector decomposition to identify the directions that capture the maximum variance within the dataset [11]. To prevent data leakage, all PCA transformation parameters, including the principal components (eigenvectors) and feature means, were fitted exclusively on the training data and subsequently applied to the validation and test sets through the same transformation process [12]. The number of retained principal components was determined based on the cumulative explained variance ratio. In Scenario B, the PCA model was configured to preserve 95% of the total explained variance [15]. The PCA formulation employed in this study follows the standard covariance matrix decomposition approach on Equation 1 [9].

$$PC_k = w_{k1}X_1 + w_{k2}X_2 + \dots + w_{kp}X_p = \sum_{j=1}^p w_{kj} X_j \quad (1)$$

In this equation, (PC_k) represents the (k) -th principal component score, (X_j) denotes the centered or standardized original variables, and (w_{kj}) signifies the eigenvectors or loading weights that maximize data variance while maintaining orthogonality.[16].

2.4. Experimental Design (3 Scenarios)

This study employed a comparative experimental design consisting of three scenarios, as presented in Table 1.

Table 1. Experimental Scenario Design

Scenario	Input	Model	Description
A	Citra 128x128x3	VGG16(Transfer Learning)	Baseline tanpa PCA
B	Vektor PCA 95%	MLP (1024-512-256-128-75)	PCA 95% explained variance
C	Vektor PCA 99%	MLP (1024-512-256-128-75)	PCA 99% explained variance

Scenario A serves as the baseline or primary comparison, using the original 128x128x3-pixel image as input, with a pre-trained VGG16 model (via transfer learning) serving as both a feature extractor and a classifier. This scenario does not use PCA, making it the benchmark for comparison against the other two scenarios. Scenario B uses feature vectors from a PCA reduction with a 95 percent explained variance threshold, which are then fed into a Multilayer Perceptron (MLP) model with a four-hidden-layer architecture of 1024-512-256-128 neurons. Scenario C uses a similar configuration but with a higher PCA threshold of 99 percent explained variance, thereby preserving more of the original feature information.

2.5. CNN Architecture

In Scenario A, a pre-trained VGG16 architecture initialized with ImageNet weights was employed as the feature extraction backbone (base model) [17]. During the initial training phase, all layers of the base model were frozen, and only the custom classifier head was trained. The classifier head consisted of a GlobalAveragePooling2D layer, followed by BatchNormalization, a Dense layer with 512 neurons and ReLU activation using L2 regularization ($\lambda = 0.0001$), a Dropout layer with a rate of 0.5, a Dense layer with 256 neurons and ReLU activation, another Dropout layer with a rate of 0.3, and a final Dense output layer with 75 neurons and Softmax activation [18].

In Scenarios B and C, a pure Multilayer Perceptron (MLP) architecture was utilized, taking PCA-transformed feature vectors as input [11]. The network architecture comprised Dense(1024), BatchNormalization, Dropout(0.5), Dense(512), BatchNormalization, Dropout(0.4), Dense(256), Dropout(0.3), Dense(128), Dropout(0.2), and a final Dense(75, Softmax) output layer. All models were trained using the Adam optimizer with an initial learning rate of 0.0001 and the categorical cross-entropy loss function.

2.6. Training Configuration

A consistent training configuration was applied across all three experimental scenarios, as summarized in Table 2.

Table 2. Training Configuration

Parameter	Value
Optimizer	Adam (lr = 0.0001)
Loss Function	Categorical Crossentropy
Batch Size	32
Max Epochs	50 (dengan EarlyStopping)
EarlyStopping	monitor = val_loss, patience = 8

Parameter	Value
ReduceLROnPlateau	factor = 0.5, patience = 4, min_lr = 1e-8
Image Size	128 × 128 × 3 piksel
Data Split	70% train / 15% val / 15% test (stratified)
Platform	Google Colab (GPU T4)
Framework	TensorFlow 2.x / Keras + Scikit-learn
Random Seed	42 (reproducibility)

The optimizer used is Adam with an initial learning rate of 0.0001. Adam is well-suited to datasets with diverse feature distributions because it adaptively adjusts the learning rate for each parameter. Categorical cross-entropy, which is the standard for multi-class classification tasks, is the loss function used. The batch size is set to 32 with a maximum of 50 epochs, but an EarlyStopping mechanism is in place to automatically halt training if the validation loss does not improve for 8 consecutive epochs [19].

In addition to EarlyStopping, this study employs ReduceLROnPlateau with a reduction factor of 0.5, a patience of 4, and a minimum learning rate threshold of 1e-8. This mechanism allows the model to adapt dynamically when the learning process begins to plateau, without halting training entirely. All images are resized to 128x128x3 pixels, and the dataset is split using a stratified strategy with 70 percent for training, 15 percent for validation, and 15 percent for testing. The experiments were run on the Google Colab platform using a T4 GPU with the TensorFlow 2.x and Keras frameworks, and a random seed of 42 to ensure the reproducibility of the results [20].

2.7. Evaluation Metrics

Two main metrics were used to assess the model's performance. Accuracy (the proportion of properly identified instances out of all samples), Precision (weighted average), Recall (weighted average), and F1-Score (weighted average), all calculated on the test dataset, were used to evaluate the first component, classification performance [21]. Training Time, measured in seconds (the total training time until the EarlyStopping criterion was met or training was finished), Inference Time, measured in milliseconds per image (calculated as the average time over 200 prediction runs on a single image), and Model Size, measured in megabytes (MB), representing the size of the saved model. The H5 model file was used to assess the second factor, computational efficiency [22].

3. RESULTS AND DISCUSSION

This section describes the experimental findings obtained from the three classification approaches developed in this research. The analysis emphasizes two key dimensions, namely classification effectiveness and computational efficiency. Classification effectiveness is measured using accuracy, precision, recall, and F1-score, whereas computational efficiency is examined through training duration, inference speed, and model complexity [21]. In addition, the influence of Principal Component Analysis (PCA) on feature dimensionality reduction, convergence behavior, and classification performance is investigated and compared with the baseline VGG16 architecture. The results are organized to provide a clear understanding of the trade-off between predictive accuracy and computational efficiency in butterfly image classification tasks.

3.1. PCA Analysis Result

The explained variance curve indicates that a substantial proportion of the dataset's variance is concentrated within a relatively small number of principal components [9]. However, retaining only 300 PCA components preserved approximately 81% of the cumulative explained variance. This finding suggests that overly aggressive dimensionality reduction may result in the loss of important visual information required to distinguish butterfly species with similar morphological characteristics. Such information loss may be one of the factors contributing to the lower classification performance of the PCA-based models compared with the baseline CNN model in this study [4], can be shown in Figures 4 and 5.

3.2. Classification Performance

Table 3 presents a comparative summary of the classification performance achieved by the three experimental scenarios on the test dataset [11].

Table 3. Comparison of Classification Performance Across All Scenario

Scenario	A: VGG16 No PCA	B: PCA-95% + MLP	C: PCA-99% + MLP
Accuracy	62.05	35.9	36.62
Precision	0.6294	0.3722	0.4023
Recall	0.6205	0.359	0.3662
F1-Score	0.6158	0.3504	0.3597
Train Time (s)	15560.8	239.1	232.1
Infer (ms)	159.36	93.49	107.34

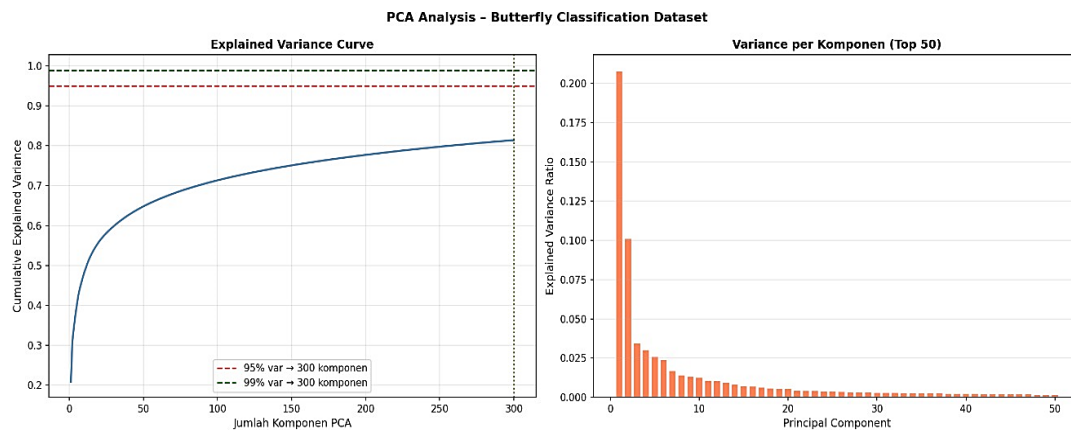


Figure 4. PCA Analysis Butterfly Classification Dataset

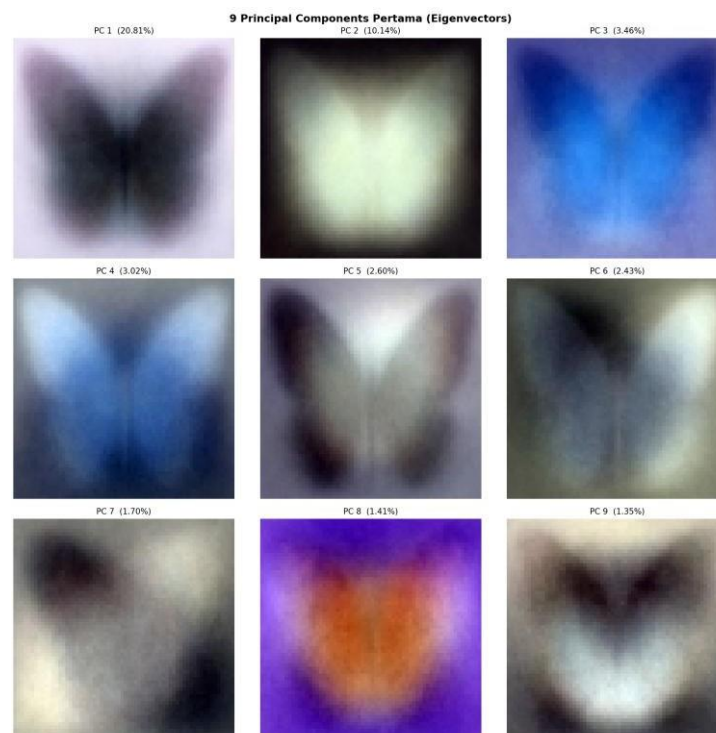


Figure 5. PCA Principal Component

Scenario A achieved the highest accuracy, 62.05 percent, on the test data. This result stems from VGG16's ability to extract rich visual feature representations through convolutional layers trained on the large-scale ImageNet dataset. Transfer learning allows the model to leverage previously learned low- and high-level features, enabling it to adapt more quickly and accurately to the 75-class butterfly dataset despite limited data availability.

Scenarios B and C show a significant drop in accuracy, achieving only 35.90 percent and 36.62 percent, respectively. The difference between Scenarios B and C is relatively small, just about 0.72 percent, indicating that increasing the PCA threshold from 95 percent to 99 percent does not yield a meaningful performance improvement. The low accuracy in the PCA- and MLP-based scenarios is likely due to the nature of the input, which has lost the spatial structure of the images after they were flattened into one-dimensional vectors prior to PCA, so that the MLP does not benefit from the inter-pixel relationships that are characteristic of image data [23].

In the context of classifying 75 butterfly species, the 62 percent accuracy achieved by Scenario A provides a sufficient baseline, given the complexity of the task and the high visual similarity among species. By comparison, Russakovsky et al. (2015) demonstrated on ImageNet that even advanced models require large amounts of data to achieve high accuracy on multi-class recognition tasks. This suggests that the results from Scenario A are already practically relevant for medium-sized datasets [24].

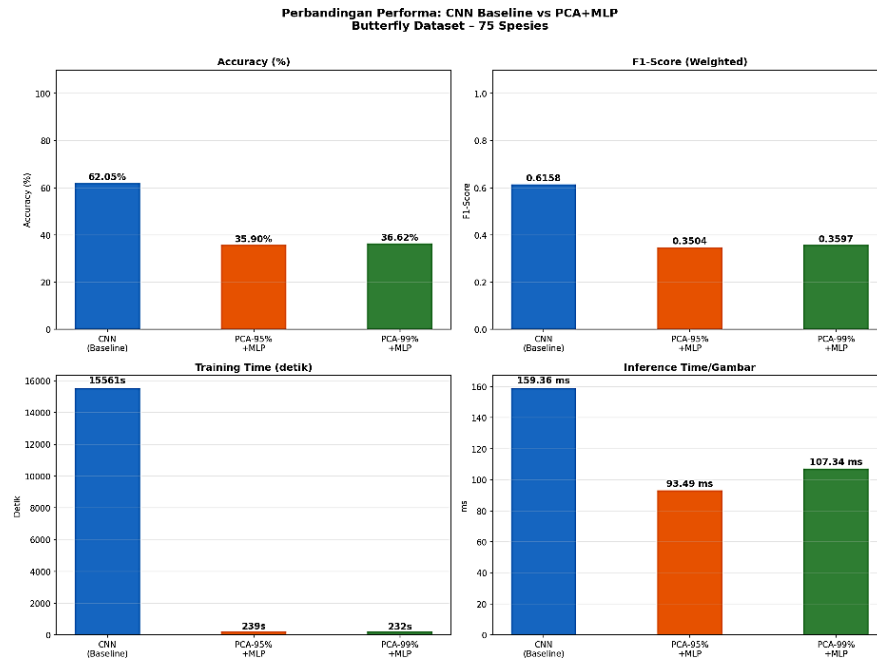


Figure 6. Performance comparison of 4 panels

The Precision value for Scenario A is 0.6294, indicating that approximately 62.94 percent of the model's predictions were correct. Meanwhile, a Recall of 0.6205 means the model successfully detected 62.05 percent of the total samples that actually belong to a specific class. The closeness of the Precision and Recall values in Scenario A indicates that the model has a good balance between avoiding false positives and false negatives, as reflected by an F1-Score of 0.6158.

In Scenarios B and C, the Precision values are 0.3722 and 0.4023, respectively, while Recall is 0.3590 and 0.3662, respectively. Although the Precision in Scenario C is slightly higher than in Scenario B, the Recall values are not significantly different; thus, the F1-Score for Scenario C (0.3597) is only slightly better than that of Scenario B (0.3504). This pattern indicates that increasing the PCA threshold from 95 percent to 99 percent does not consistently improve the model's ability to classify all classes evenly.

The difference in F1-Score between Scenario A and Scenarios B and C, which reaches nearly 0.26 points, is a substantial and significant gap [25]. This reinforces the argument that CNN-based feature representations are far more effective than PCA-based representations for multi-class image classification tasks with high visual variation between classes. This finding is consistent with the results of the study by Alzubaidi et al. (2021), which emphasizes that CNNs are inherently superior at handling image data compared to linear projection-based methods such as PCA [26].

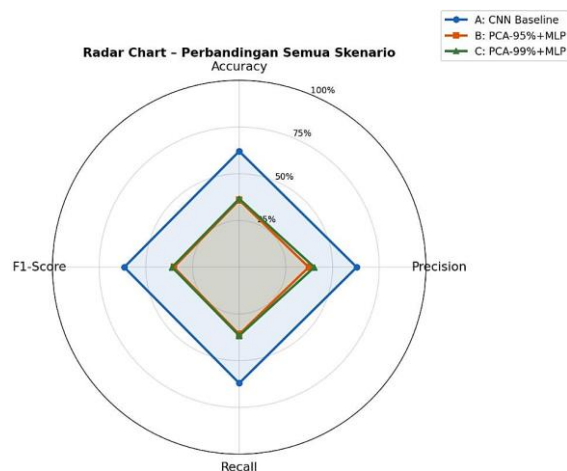


Figure 7. Radar Chart

3.3. Training History

The three experimental settings show different convergence characteristics in their training and validation accuracy/loss curves. Training accuracy increased steadily during the training phase in Scenario A

(VGG16 without PCA), reaching nearly 80% by the final epoch [27]. In the meantime, validation accuracy increased to between 58% and 60% and started to level out around the 25th epoch. Effective learning was demonstrated by the training loss and validation loss curves, which both steadily decreased over time. Nonetheless, minor overfitting may be present given the growing discrepancy between training and validation accuracy in the latter epochs [28].

In Scenario B (PCA 95% + MLP), training accuracy increased to approximately 52%, whereas validation accuracy reached only around 35%–36%. Although the loss curves continued to decrease, improvements in validation accuracy became less pronounced after the 30th epoch. This behavior indicates that the model learned patterns from the training data but exhibited limited generalization to unseen validation samples. Similarly, Scenario C (PCA 99% + MLP) demonstrated a convergence pattern comparable to that of Scenario B. Training accuracy reached approximately 54%, while validation accuracy remained within the range of 36%–37%. The slightly higher validation accuracy achieved by Scenario C suggests that retaining a greater proportion of the original variance through PCA can contribute to improved classification performance. All three scenarios were trained for the full 50 epochs because the EarlyStopping mechanism was not activated during training, which can be viewed in Figures 8–10.

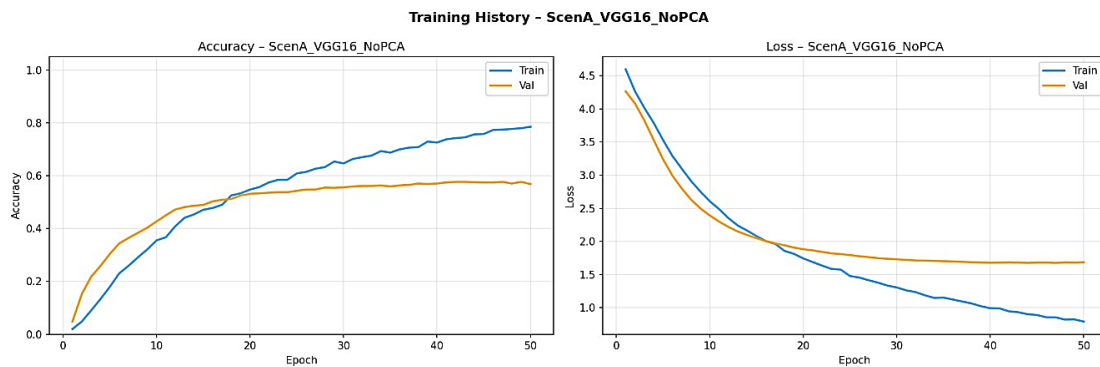


Figure 8. Training - Scen A VGG16 No PCA

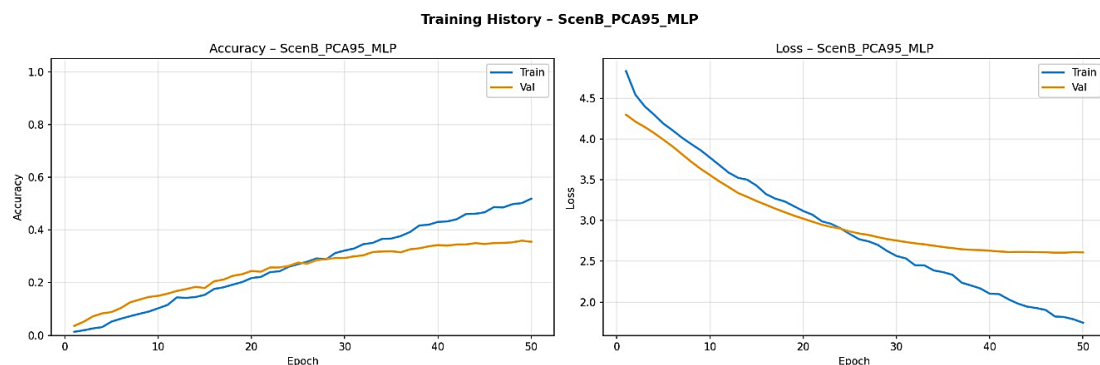


Figure 9. Training Scen B PCA95+MLP



Figure 10. Training- ScenC PCA99+MLP

Overall, the differences in convergence behavior indicate that PCA-based dimensionality reduction effectively decreases data complexity and accelerates computational processes. However, it may also remove important visual information required to distinguish subtle characteristics among butterfly species. As a result,

computational efficiency came at the expense of classification performance. The VGG16 model consistently achieved the highest accuracy of 62.05%, outperforming PCA95+MLP and PCA99+MLP, which obtained accuracies of 35.90% and 36.62%, respectively (Table 4).

Table 4. Final Comparison

Scenario	A: VGG16 No PCA	B: PCA 95% + MLP	B: PCA 95% + MLP
Input Dim	12288	300	300
Accuracy (%)	62.05	35.9	36.62
Best Val (%)	57.64	35.9	36.21
Precision	0.6294	0.3722	0.4023
Recall	0.6205	0.359	0.3662
F1-Score	0.6158	0.3504	0.3597
Train Time	15560.8	239.1	232.1
Epochs	50	50	50
Infer (ms)	159.36	93.49	107.34
Model (MB)	60.97	11.63	11.63
Params	15,129,995	1,013,067	1,013,067

3.6. Discussions

The experimental results reveal several important insights that merit detailed discussion. Scenario A (VGG16 without PCA) achieved the highest accuracy of 62.05%, outperforming Scenarios B and C by more than 26 percentage points. This accuracy gap is primarily attributable to the fundamental architectural difference between the models: VGG16 leverages convolutional layers pre-trained on ImageNet to extract rich spatial hierarchical features from pixel-level inputs, whereas the MLP classifiers in Scenarios B and C receive PCA-transformed vectors that, while mathematically compact, discard spatial structure inherent to image data. PCA operates on flattened pixel vectors and therefore cannot preserve the spatial locality that CNN convolutional filters exploit. This finding aligns with Zhao et al. (2022), who demonstrated that naive flattening followed by dimensionality reduction prior to classification incurs substantial information loss compared to pooling strategies embedded within CNN architectures [14].

The dimensionality reduction achieved by PCA was substantial, compressing 12,288 input dimensions to only 300 principal components (a 97.6% reduction), yet retaining only limited discriminative spatial information. This explains the convergence plateau observed in Scenarios B and C, where validation accuracy stagnated at approximately 35-36% despite a continued decrease in training loss, indicating persistent underfitting rather than overfitting. From a computational efficiency perspective, PCA delivered exceptional gains: training time was reduced from 15,560.8 seconds to 239.1 seconds in Scenario B (98.5% reduction) and 232.1 seconds in Scenario C. Inference time decreased from 159.36 ms to 93.49 ms and 107.34 ms, respectively. Model size was reduced from 60.97 MB (VGG16) to 11.63 MB for both PCA-based models. These computational gains are consistent with findings by Buatoom and Jamil (2023) and Liu et al. (2025), who demonstrated that dimensionality reduction substantially lowers computational demands in image classification pipelines [13], [15].

The performance trade-off identified in this study suggests that PCA-based preprocessing is better suited for resource-constrained deployment scenarios where speed and model size are prioritized over peak accuracy, rather than high-accuracy research applications. A more promising direction for combining PCA and CNN would be to apply PCA after CNN feature extraction (i.e., on the flattened feature map from the final convolutional layer) rather than on raw pixel inputs, thereby preserving spatial features while still achieving dimensionality reduction before the classifier head. This approach is recommended for future work.

4. CONCLUSION

This study aimed to investigate the effect of applying Principal Component Analysis (PCA) as a dimensionality-reduction technique on the performance of a Convolutional Neural Network (CNN) for classifying images of 75 butterfly species, using the phucthaiv02 Butterfly Image Classification dataset from Kaggle. To achieve this objective, three experimental scenarios were developed. Scenario A employed a transfer-learning-based VGG16 architecture as the baseline model, without PCA. Scenario B applied PCA with a 95% explained variance threshold prior to classification using a Multilayer Perceptron (MLP), while Scenario C implemented PCA with a 99% explained variance threshold.

The experimental results indicate that Scenario A achieved the highest classification performance, obtaining an accuracy of 62.05% and an F1-score of 0.6158. In comparison, Scenario B and Scenario C achieved accuracies of 35.90% and 36.62%, respectively. Although the integration of PCA reduced classification accuracy by approximately 26 percentage points relative to the baseline model, it substantially improved computational efficiency. Training time decreased from 15,560.8 seconds for the VGG16 model to 239.1 seconds in Scenario B and 232.1 seconds in Scenario C, representing a reduction of approximately 98.5%. Furthermore, inference latency was reduced from 159.36 ms to 93.49 ms and 107.34 ms, respectively.

These findings suggest that PCA is highly effective in reducing computational costs; however, this advantage is accompanied by a noticeable decline in classification performance.

Several limitations of this study should be acknowledged. First, the pre-trained VGG16 model was not fine-tuned; unfreezing and retraining the final convolutional blocks may improve baseline accuracy. Second, only a single CNN architecture (VGG16) and a single MLP architecture were evaluated; future work should explore ResNet50, EfficientNet, and Vision Transformers as alternative backbones. Third, data augmentation was not applied, which may have limited generalization. Fourth, PCA was applied exclusively as a preprocessing step to raw pixel inputs; a more effective approach may be to apply PCA to CNN-extracted feature maps, preserving spatial information while reducing the classifier's input dimensionality. From a practical standpoint, the findings suggest that VGG16 transfer learning without PCA is recommended when classification accuracy is the primary concern, whereas PCA-MLP pipelines are more appropriate for edge deployment or systems with strict computational constraints (98.5% reduction in training time and 80.9% reduction in model size). Future research is encouraged to investigate: (1) applying PCA after CNN feature extraction rather than on raw pixels; (2) combining PCA with data augmentation and fine-tuning; (3) comparing PCA with other dimensionality reduction methods such as autoencoders or t-SNE, for feature compression; and (4) evaluating these pipelines on larger, more diverse butterfly datasets to improve the generalizability of the findings.

REFERENCES

- [1] H. Anwar, "Butterflies and benchmarks: a systematic evaluation of deep learning backbones to identify the best model for image-based butterflies species recognition," *Pattern Anal. Appl.*, vol. 29, no. 1, p. 13, Mar. 2026, doi: 10.1007/s10044-025-01587-7.
- [2] F. Rajeeva P. P. et al., "A Novel Method for the Classification of Butterfly Species Using Pre-Trained CNN Models," *Electronics*, vol. 11, no. 13, p. 2016, Jun. 2022, doi: 10.3390/electronics11132016.
- [3] R. F. Mattins, M. V. R. Sarobin, A. A. Aziz, and S. Srivarshan, "Object detection and classification of butterflies using efficient CNN and pre-trained deep convolutional neural networks," *Multimed. Tools Appl.*, vol. 83, no. 16, pp. 48457–48482, Nov. 2023, doi: 10.1007/s11042-023-17563-4.
- [4] M. Teke and G. Elsamoly, "Innovative deep learning solutions for Turkish butterfly species identification: a VGGNet enhancement study," *Signal, Image Video Process.*, vol. 19, no. 5, p. 396, May 2025, doi: 10.1007/s11760-025-03979-y.
- [5] N. T. Sarshar, S. Sadeghi, M. Kamsari, M. Avazpour, S. J. Ghouschi, and R. Ranjbarzadeh, "Advancing Brain MRI Image Classification: Integrating VGG16 and ResNet50 with a Multi-Verse Optimization Method," *BioMed*, vol. 4, no. 4, pp. 499–523, Nov. 2024, doi: 10.3390/biomed4040038.
- [6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
- [7] A. W. Salehi et al., "A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope," *Sustainability*, vol. 15, no. 7, p. 5930, Mar. 2023, doi: 10.3390/su15075930.
- [8] D. Satria Yudha Kartika and H. Maulana, "Preprocessing dan normalisasi pada dataset kupu-kupu untuk ekstraksi fitur warna, bentuk dan tekstur," *J. Comput. Electron. Telecommun.*, vol. 1, no. 2, Jan. 2021, doi: 10.52435/complete.v1i2.76.
- [9] Nadiyah Hidayati and M. Maulidah, "Ekstraksi Fitur Dengan Color Histogram Dan Classifier Random Forest Pada Citra Kupu-Kupu," *JAMI J. Ahli Muda Indones.*, vol. 4, no. 2, pp. 148–157, Dec. 2023, doi: 10.46510/jami.v4i2.172.
- [10] H. Liu, W. Bi, and N. Mughees, "Enhanced Hyperspectral Image Classification Technique Using PCA-2D-CNN Algorithm and Null Spectrum Hyperpixel Features," *Sensors*, vol. 25, no. 18, Sep. 2025, doi: 10.3390/s25185790.
- [11] R. Madinakhon, D. Mukhtorov, and Y.-I. Cho, "Integrating Principal Component Analysis and Multi-Input Convolutional Neural Networks for Advanced Skin Lesion Cancer Classification," *Appl. Sci.*, vol. 14, no. 12, p. 5233, Jun. 2024, doi: 10.3390/app14125233.
- [12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.
- [13] U. Buatoom and M. U. Jamil, "Improving Classification Performance with Statistically Weighted Dimensions and Dimensionality Reduction," *Appl. Sci.*, vol. 13, no. 3, p. 2005, Feb. 2023, doi: 10.3390/app13032005.
- [14] B. Zhao, X. Dong, Y. Guo, X. Jia, and Y. Huang, "PCA Dimensionality Reduction Method for Image Classification," *Neural Process. Lett.*, vol. 54, no. 1, pp. 347–368, Feb. 2022, doi: 10.1007/s11063-021-10632-5.
- [15] I. Garg, P. Panda, and K. Roy, "A Low Effort Approach to Structured CNN Design Using PCA," *IEEE Access*, vol. 8, pp. 1347–1360, 2020, doi: 10.1109/ACCESS.2019.2961960.
- [16] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A Comprehensive Review of

- Dimensionality Reduction Techniques for Feature Selection and Feature Extraction,” *J. Appl. Sci. Technol. Trends*, vol. 1, no. 1, pp. 56–70, May 2020, doi: 10.38094/jastt1224.
- [17] C. Xu et al., “Fine-Grained Butterfly Recognition via Peer Learning Network with Distribution-Aware Penalty Mechanism,” *Animals*, vol. 12, no. 20, p. 2884, Oct. 2022, doi: 10.3390/ani12202884.
- [18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.
- [19] Y. Liu and L. J. Durlinsky, “3D CNN-PCA: A Deep-Learning-Based Parameterization for Complex Geomodels,” Jul. 2020, doi: 10.1016/j.cageo.2020.104676.
- [20] A. A. Wani, “Comprehensive review of dimensionality reduction algorithms: challenges, limitations, and innovative solutions,” *PeerJ Comput. Sci.*, vol. 11, p. e3025, Jul. 2025, doi: 10.7717/peerj-cs.3025.
- [21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 2818–2826. doi: 10.1109/CVPR.2016.308.
- [22] N. Chothani, I. Desai, C. K. chan, S. Singh, D. Varshney, and N. Naik, “PCA and CNN-based detection and classification of faults in distribution network with distributed energy resources,” *Appl. Eng. Sci.*, vol. 25, p. 100300, Mar. 2026, doi: 10.1016/j.apples.2026.100300.
- [23] A. Mardiansyah, E. C. Djamal, and F. Nugraha, “Multivariate EEG Signal Using PCA and CNN in Post-Stroke Classification,” in *2020 FORTEI-International Conference on Electrical Engineering (FORTEI-ICEE)*, IEEE, Sep. 2020, pp. 113–118. doi: 10.1109/FORTEI-ICEE50915.2020.9249880.
- [24] O. Russakovsky et al., “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.
- [25] D. Coelho, A. Madureira, I. Pereira, and R. Gonçalves, “A Review on Dimensionality Reduction for Machine Learning,” 2023, pp. 287–296. doi: 10.1007/978-3-031-27499-2_27.
- [26] L. Alzubaidi et al., “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *J. Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.
- [27] D. Anggreani, “Peningkatan Metode YOLOv7 Dengan Proses Augmentasi Image Pada Klasifikasi Jenis Kupu-Kupu YOLOv7 Method Improvement With Image Augmentation Process In Classification of Butterfly species,” 2023.
- [28] R. Li, H. Liu, and Y. Hu, “Financial performance evaluation and forecasting of enterprises by the combination of PCA and CNN deep learning,” Dec. 01, 2023, Higher Education Press Limited Company. doi: 10.1007/s11704-023-2494-1.
- [29] A. Maftukhah, A. Fadlil, and S. Sunardi, “Butterfly Image Classification using Convolution Neural Network with AlexNet Architecture,” *J. INFOTEL*, vol. 16, no. 1, Feb. 2024, doi: 10.20895/infotel.v16i1.1004.
- [30] E. Hartati, K. Kunci, and K. Kupu, MDP Student Conference (MSC) 2022 Klasifikasi Spesies Kupu Kupu Menggunakan Metode Convolutional Neural Network.