Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th February 2025. Vol. 4. No. 2; e-ISSN: 2808-4519

Implementation of MobileNet V3 In Classifying Butterfly Species with Android and Cloud Based Application Development

Ihsan Zulfahmi[1]*, Said Iskandar Al Idrus[2], Hermawan Syahputra[3], Insan Taufik[4], Kana Saputra S[5]

[1,2,3,4]*Ilmu Komputer, Universitas Negeri Medan*
[5]*Matematika, Universitas Negeri Medan*
ihsanzulfahmi21@gmail.com [1*]

Abstract

This research aimed to develop an Android application capable of classifying butterfly species using cloud computing and deep learning technologies. MobileNetV3-Large, a Convolutional Neural Network (CNN) architecture, was employed to process and classify six butterfly species. The dataset was divided into two ratios, 70:30 and 80:20, for training and testing. Evaluation results indicated that the optimal model was achieved with an 80:20 ratio, yielding an accuracy of 94% and precision, recall, and F1-Score values exceeding 90% for each species class. Google Cloud Platform (GCP) was utilized to manage and run the model using the Cloud Run service, enabling the application to function efficiently even with limited resources on Android devices. The application incorporates an encyclopedia of species and a camera scanning feature, making it a valuable educational tool

Keywords: *MobileNetV3-Large, Butterfly, CNN, GCP, Cloud Run, Android, Deep Learning*

1. Introduction

Butterflies, insects from the order *Lepidoptera*, are vital to ecosystems as pollinators, facilitating plant reproduction, and serving as bioindicators of environmental health due to their observable traits, such as vibrant wing patterns, slow flight, and diurnal activity (Lamatoa et al., 2013; Pe'er & Settele, 2008). However, their populations are highly sensitive to environmental changes, with habitat loss, pollution, and land-use changes significantly impacting species diversity and abundance (Firmalinda, 2007).

Indonesia, home to approximately 2,000 butterfly species in 2020, benefits from its tropical climate conducive to butterfly proliferation. Unfortunately, urban development, deforestation, and environmental degradation have led to a decline in many species, some of which are now rare. This loss highlights the urgent need for practical tools to identify and monitor butterfly species to support conservation efforts.

Despite their ecological importance, there is limited access to user-friendly tools for butterfly species identification. This gap poses challenges for researchers, educators, and the general public in obtaining accurate and timely information, particularly for monitoring endangered species. Existing web-based butterfly classification tools often lack cloud computing integration, limiting data accessibility and scalability, and are suboptimal for mobile platforms.

To address these limitations, this study proposes the development of an Android-based application utilizing deep learning and cloud computing. The proposed system employs MobileNetV3-Large, a lightweight and efficient convolutional neural network (CNN) architecture suitable for mobile devices, to extract features from butterfly images and perform accurate species classification (Qian et al., 2021). Integrating cloud computing ensures efficient data storage and accessibility, enabling a robust tool for butterfly conservation and biodiversity preservation in Indonesia. Sample Selection

This study focuses on six butterfly species as the primary samples for testing and analysis: *Boolina eggfly*, *Catopsilia Scylla*, *Graphium Sarpedon*, *Peacock*, *Papilio memnon*, and *Samia Ricini*. These species were selected to ensure a diverse representation in the dataset, enabling robust classification and analysis.

2. Research Method

This study comprises seven processes (as illustrated in Figure 1):

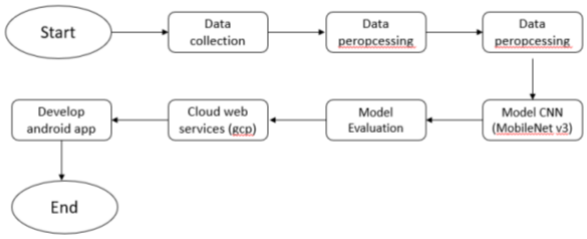

Fig. 1: Research workflow

2.1. Data Collection

Data Collection

This study utilized both primary and secondary data collection methods. A total of 50 primary samples were obtained through direct field observations to ensure accuracy and relevance of the analyzed butterfly species. Additionally, 60 secondary samples were sourced from documented literature and existing butterfly collections, enriching the dataset. The combined dataset, comprising 110 samples across six butterfly species, was augmented using random image augmentation techniques, generating 1,000 additional samples per class. This process resulted in a total of 6,000 augmented images, enhancing feature diversity, mitigating overfitting, and improving model generalization, as shown in Table 1.

Table 1: Comparison of total data classes

Label	Kelas Spesies	Jumlah
0	Boolina eggfly	1000
1	Catopsilia Scylla	1000
2	Graphium Sarpedon	1000
3	Peacock	1000
4	Papilio memnon r	1000
5	Samia Ricini	1000

2.2. Data Preprocessing

After preparing the dataset, the data preprocessing stage was carried out to ensure data quality and consistency. This process involved verifying the alignment of images with their respective folders, labeling each data point to identify the classes for the classification task, and resizing images to maintain uniform dimensions. Additionally, data augmentation was applied to the training images to enhance model performance and reduce the risk of overfitting.

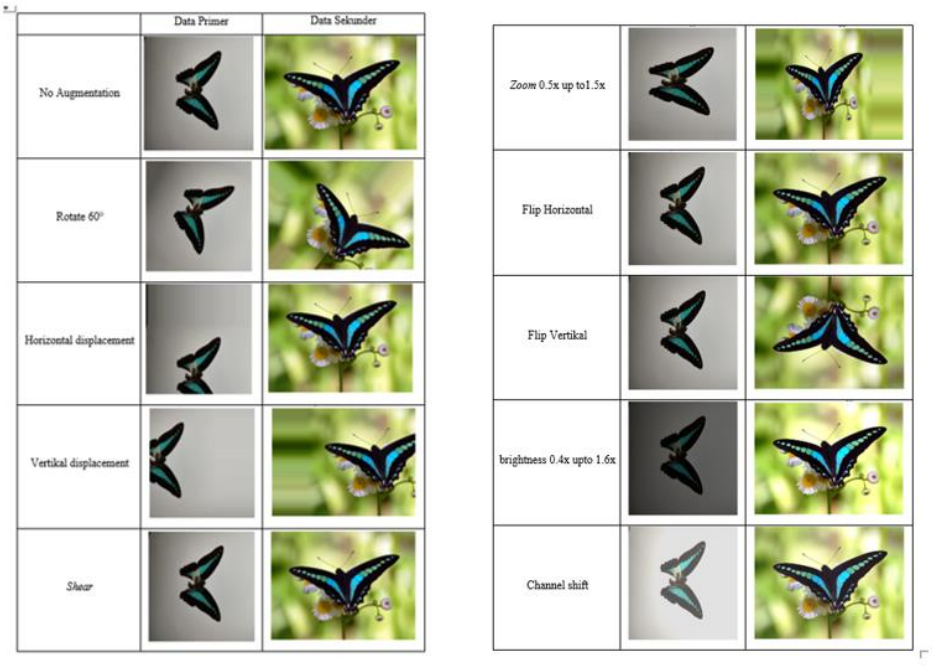

Fig. 2: Results of image augmentation

2.3. Split Dataset

The dataset was divided into two subsets: training data and testing data, using two different split ratios: 70:30 and 80:20. In the 70:30 split, the dataset consisted of 4,200 images for training and 1,800 images for testing. Meanwhile, in the 80:20 split, 4,800 images were used for training and 1,200 for testing. The *train_test_split* method was employed to ensure a well-balanced distribution of each class across both subsets.

2.4. Model CNN

In this stage, a CNN model was constructed by adding a fully connected layer to the pre-defined Transfer Learning models. The hyperparameters used to build the CNN model are outlined in Table 2.

Table 2: Hyperparameters of the CNN Model

Hyperparameter	Nilai
Optimization	Adam
Learning Rate	1×10^{-4}
Epoch	40
Batch	32
Loss function	Sparse Categorical Crossentropy
Metrics	Acurration

The optimization algorithm used in this study was Adaptive Momentum Optimization (Adam), selected for its superior validation accuracy and lower loss compared to other optimizers such as Rprop, Adagrad, Adamax, and Nadam, with a learning rate of 1×10^{-4} [16]. The CNN model was trained on preprocessed data using a batch size of 32 and 40 epochs, which has been shown in previous studies to yield optimal accuracy [17]. Transfer Learning was applied using the MobileNetV3 architecture, which consists of two primary stages: feature learning and classification. The feature learning stage leverages convolutional layers to identify image patterns, while the classification stage uses these patterns to predict the class labels.

2.5. Model Evaluation

The model evaluation stage involved calculating metrics such as confusion matrix, recall, precision, F1 score, and accuracy to comprehensively assess performance. By monitoring the number of true positives, true negatives, false positives, and false negatives, these metrics were computed using Equations 1 through 4 [18].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1\ Score = 2 \cdot \frac{1}{\frac{1}{Precision}+\frac{1}{Recall}} \quad (4)$$

2.6. Web Services Google Cloud Platform (GCP)

Google Cloud Platform (GCP) offers comprehensive solutions for deploying and managing applications, including Virtual Machines (VMs) and Cloud Run. VMs, available through Compute Engine, provide highly customizable and scalable compute resources, allowing users to run various workloads, from simple applications to complex machine learning models. VMs offer flexibility in terms of operating systems, hardware configurations, and network management, making them suitable for diverse use cases.

On the other hand, Cloud Run enables serverless application deployment, specifically for containerized workloads. It eliminates the need to manage servers by automatically scaling resources based on demand, ensuring optimal performance and cost-efficiency. With Cloud Run, developers can deploy applications quickly using container images, benefiting from seamless integration with other GCP services such as Cloud Storage and Cloud SQL. Together, these features make GCP an ideal platform for both traditional infrastructure needs and modern, cloud-native application development.

2.7. Develop Android App

Developing Android applications involves creating software for mobile devices using the Android operating system. It requires utilizing tools such as Android Studio, Java, and Kotlin for building user-friendly interfaces and integrating backend services. The development process includes stages such as design, coding, testing, and deployment, ensuring functionality, performance, and usability. Android development also requires considering device compatibility, optimizing resource usage, and leveraging Android-specific features like push notifications, location services, and camera functionalities to enhance user experience.

3. Experiments and Analysis

3.1. Experimental setup

This research designs a system using programming language and hardware according to needs, with the aim of classifying butterfly species using the CCN algorithm on the MobileNet V3 architecture. The hardware and software components used are:

Table 3: Setup Spesification

Hardware	Spesification
• Laptop	• Prosesor: Intel Core i5 – 8520U @ 1.8 Ghz. (8 CPU) • RAM: Physical memory 12 GB • Sistem Operasi: Windows (64-bit operating system, x64-based processor) • VGA Nvidia MX150 2 GB • 256 SSD and 1 TB HDD
• Smartphone	• Model: Samsung Galaxy A32 • Sistem Operasi: One UI 5.0, Android 13 • Kamera: 64 MP, f/1.8, 26mm (wide), PDAF • Mediatek Dimensity 720 (7 nm) • Octa-core (2x2.0 GHz Cortex-A76 & 6x2.0 GHz Cortex-A55) • Mali-G57 MC3

SOFTWARE	
• Windows 11 Pro (64-bit)	• TensorFlow
• Microsoft Edge	• Android Studio
• Visual Studio Code	• Docker
• Google Colaboratory	• Google Cloud Platform (GCP)
• Python 3.12.4	• Google Drive

3.2. classification model results

The experimental evaluation of the butterfly species classification model was conducted utilizing the MobileNetV3-Large architecture. Two distinct data partition scenarios were implemented: a 70:30 and an 80:20 ratio for training and validation sets, respectively. The primary objective of this evaluation was to assess the model's classification performance across six butterfly species while optimizing accuracy metrics. The evaluation framework incorporated multiple performance metrics, including Precision, Recall, F1-Score, and Validation Accuracy, which were calculated for each species class. The comprehensive results of this evaluation are presented in Table

Table 4: Model Result

Model	*Rasio*	Kelas Data	*Precision*	*Recall*	*F1 score*	*Validation Accuracy* (%)
MobileNet V3 Large	70:30	Boolina eggfly	0.88	0.89	0.89	90
		Catopsilia Scylla	0.96	0.87	0.91	
		Graphium Sarpedon	0.86	0.86	0.86	
		Peacock	0.96	0.94	0.95	
		Papilio memnon r	0.85	0.90	0.88	
		Samia Ricini	0.91	0.95	0.93	
	80:20	Boolina eggfly	0.91	0.95	0.93	94
		Catopsilia Scylla	0.98	0.94	0.96	
		Graphium Sarpedon	0.89	0.90	0.89	
		Peacock	0.98	0.97	0.98	
		Papilio memnon r	0.93	0.92	0.92	
		Samia Ricini	0.94	0.95	0.95	

Based on the tabulated results, the 80:20 data ratio demonstrated superior performance compared to the 70:30 ratio, evidenced by improved Precision, Recall, and F1-Score values across most butterfly species classes. Notably, Boolina eggfly, Peacock, and Samia Ricini exhibited significant performance enhancements at the 80:20 ratio, achieving an overall validation accuracy of 94%. This suggests that the increased training data volume enabled more effective pattern recognition and enhanced classification accuracy.

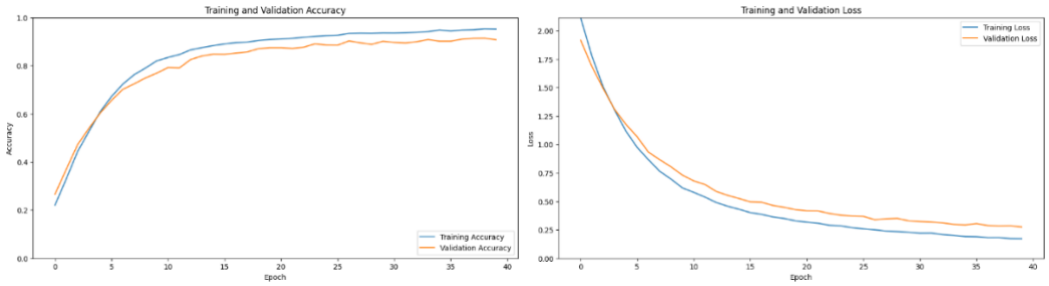

Fig. 3: Accuracy and loss graph with a 70 : 30 division

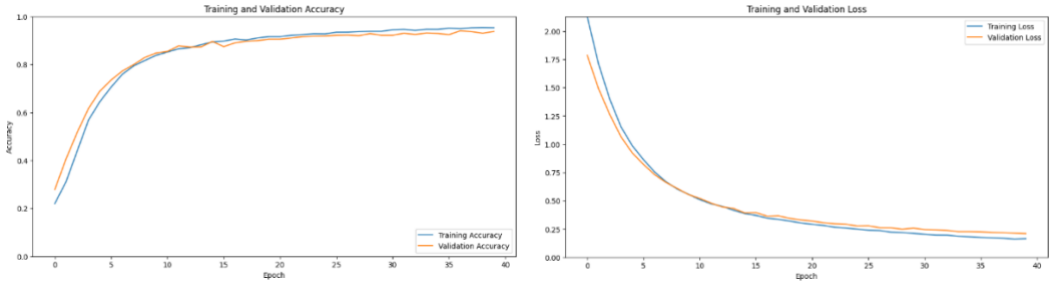

Fig. 4: Accuracy and loss graph with 80:20 data division

The accuracy and loss graphs demonstrate consistent progression throughout the training process. For the 70:30 ratio, both training and validation accuracy increased sharply during the initial 15 epochs, stabilizing around 90% after approximately 35 epochs. In contrast, the 80:20 ratio achieved superior accuracy, approaching 94% at training completion. This indicates that increased training data volume contributes to more stable and accurate model performance. The loss graphs for both ratios exhibit steady decrements as epochs progress, with the 80:20 ratio showing lower loss values, indicating enhanced error minimization during both training and validation phases

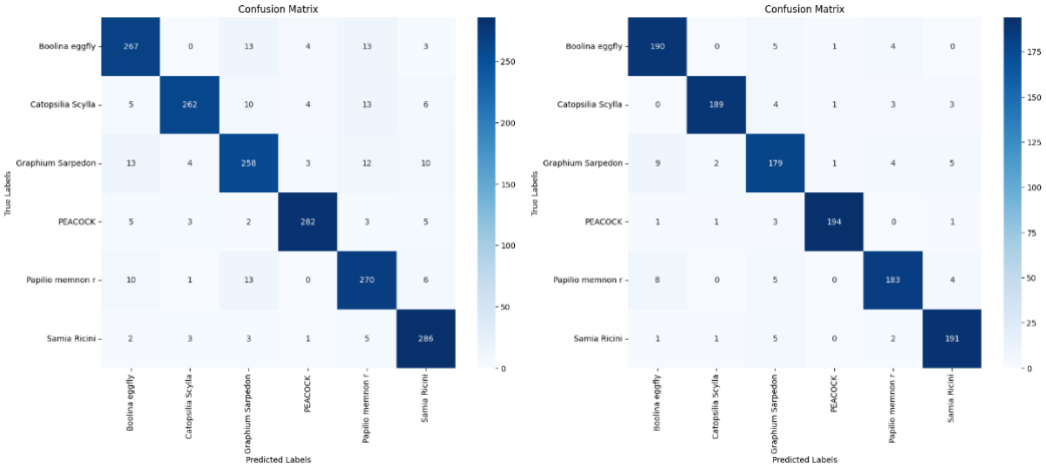

Fig. 5: Confusion matrix with split 70:30 (left) dan 80:20 (Right)

The confusion matrix provides further insights into the model's performance in butterfly species classification. In the 70:30 ratio, classification errors were observed for Graphium Sarpedon, frequently misclassified as Boolina eggfly and Papilio memnon r. However, the majority of predictions align along the diagonal matrix, indicating successful predictions for most validation data. The 80:20 ratio demonstrates superior performance with fewer classification errors, suggesting that MobileNetV3-Large achieves higher prediction accuracy with a larger training data proportion.

Based on the evaluation of MobileNetV3-Large across both data partition scenarios (70:30 and 80:20), the 80:20 ratio exhibits superior performance, evidenced by:

1. Increased validation accuracy reaching 94%
2. Enhanced evaluation metrics (Precision, Recall, and F1-Score) across most butterfly species classes
3. More stable accuracy and loss progression compared to the 70:30 ratio

Consequently, the MobileNetV3-Large architecture with an 80:20 data ratio was selected for implementation in the Android and Cloud-based butterfly species classification application.

3.3. Development of GCP (Google Cloud Platform) Web Services

The web services implementation leverages two primary features of Google Cloud Platform (GCP): Virtual Machine (VM) and Cloud Run. These features play crucial roles in supporting the application architecture, operating independently while maintaining integration.

3.3.1 Virtual Machine (VM) Implementation

Virtual Machine serves as the primary infrastructure for executing JavaScript-based application components. In this context, VM functions as a server hosting critical services, including Login and Registration systems, Species and Butterfly Family Libraries, operating under a pay-as-you-go model.The deployment of a VM instance on GCP generates an external IP endpoint for Android application connectivity. This endpoint facilitates HTTP protocol-based function execution for the Android application. JavaScript files (.js) define command handlers for requests and responses at each endpoint. Table 4.4 presents the API endpoints with their respective HTTP methods and descriptions:

Table 5: Endpoit Result

no	Endpoint	metode	deskripsi
1	http://35.243.66.140:8000/Login	POST	Menangani proses autentikasi pengguna pada aplikasi
2	http://35.243.66.140:8000/Register	POST	Menangani proses pendaftaran akun pengguna
3	http://35.243.66.140:8000/family	GET	Menampilkan daftar family pada aplikasi
4	http://35.243.66.140:8000/Species	GET	Menampilkan daftar species pada aplikasi

These endpoints are hosted via GCP's Virtual Machine service, utilizing external IP protocols for Android-server communication through the Application Programming Interface (API), enabling real-time backend interaction.

3.3.2 Cloud Run Implementation

Cloud Run's primary function is executing containerized applications in a serverless environment. It processes machine learning results, serving as the application's core intelligence by storing and executing machine learning models online. The implementation requires several key files:
- butterfly.keras: Trained machine learning model
- app.py: Flask API application connecting model and users
- Dockerfile: Container build instructions
- requirements.txt: Required Python dependencies
- Message.json: Model scanning result output configuration

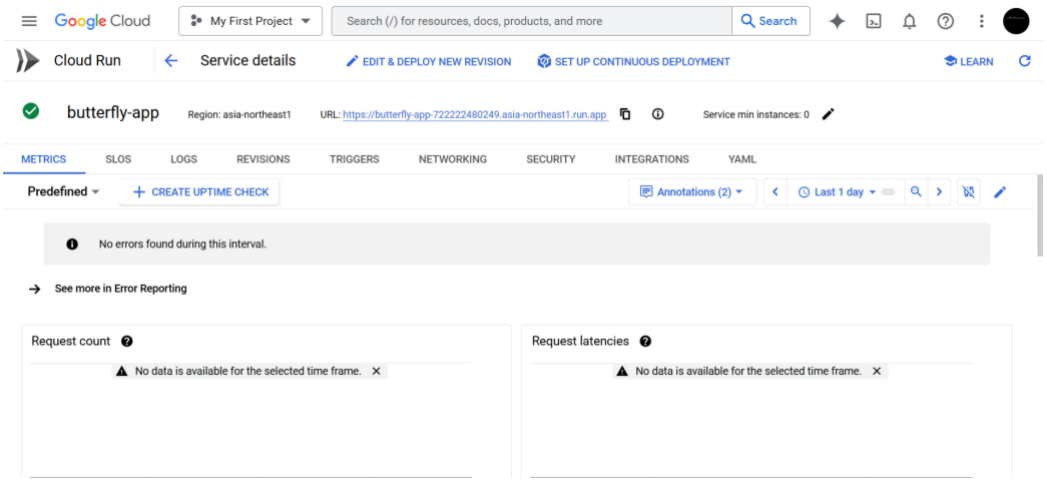

Fig. 6: Cloud Run Dashboard and Application Endpoints

The deployed endpoint (https://butterfly-app-722222480249.asia-northeast1.run.app/classify) facilitates inter-program communication for Android application integration. Initial endpoint testing via Postman confirmed proper functionality, with message outputs configurable through app.py.

3.4 Application Development

3.4.1 Model Conversion

The model, initially stored in ".h5" format, required conversion to ".keras" format for Android compatibility. The conversion was performed on the best-performing model (80:20 ratio) as determined by the evaluation metrics in Table 4.

3.4.2 Interface Design

The application architecture comprises two primary interfaces:
Main Interfaces:
- Home Page: Houses the butterfly species encyclopedia and family database

- Scan Page: Provides butterfly object recognition functionality through camera capture or gallery upload

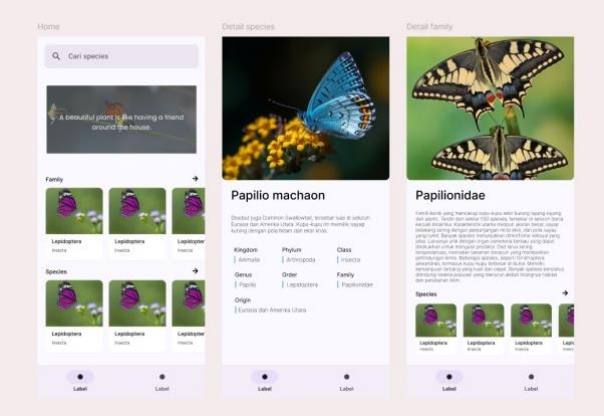

Fig. 7: Library display on the home page

Authentication Interfaces:

- Login Page: Controls access to registered users
- Registration Page: Facilitates new user account creation

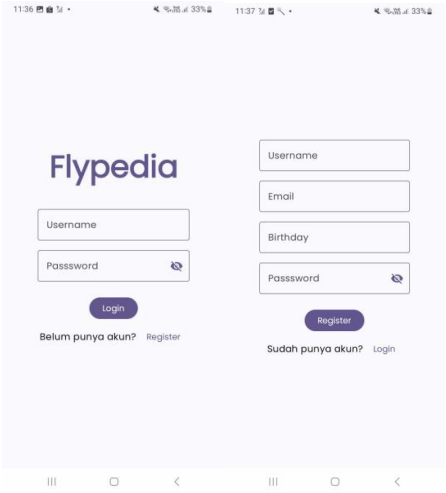

Fig. 8: Display login and registration pages

3.4.3 Design Implementation

The implementation encompasses four key components:

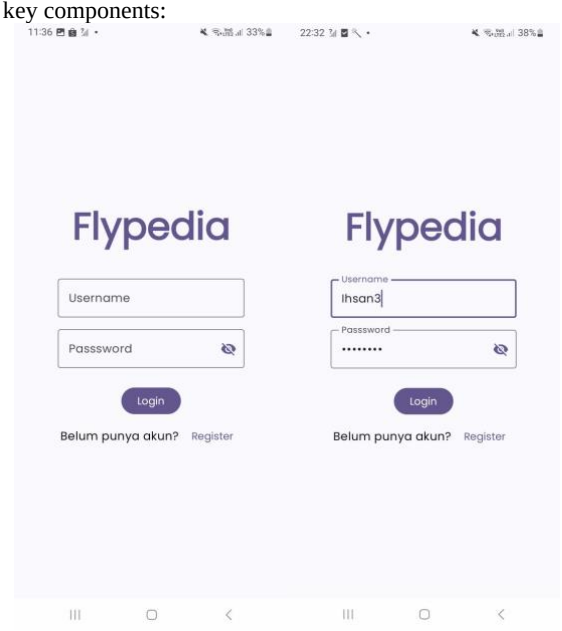

Fig. 9: Login page display design

Authentication System
- Login interface: Initial access point requiring username and password
- Registration interface: New user account creation with email validation
- Server-side API integration for account verification

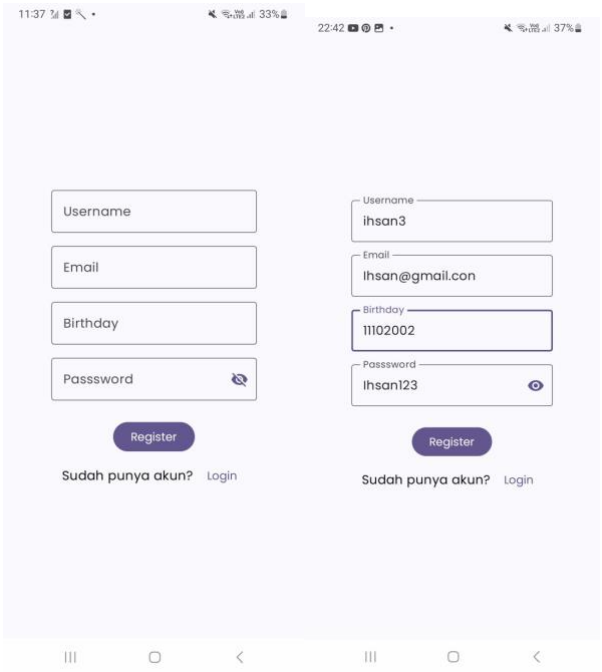

Fig. 10: Registration page display design

Home Interface
- Encyclopedia interface displaying butterfly species and families
- Interactive elements for detailed species information
- Family classification browsing functionality

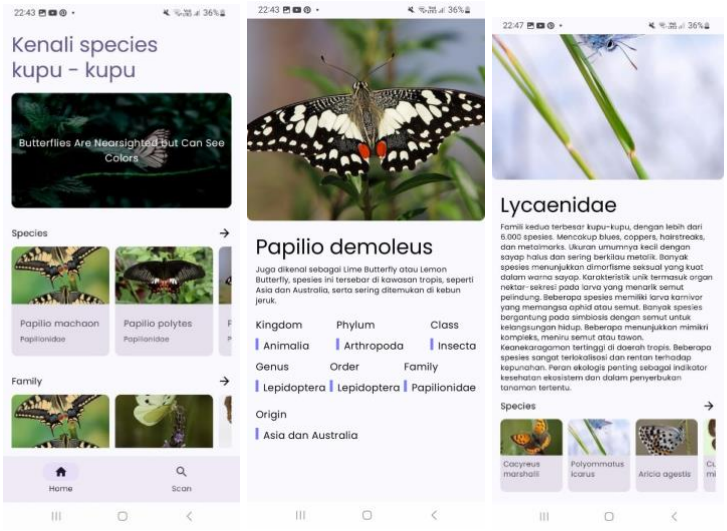

Fig. 11: Home page display design

Scanning Interface
- Dual capture options:
- Direct camera capture
- Gallery image upload
- Online species classification
- Detailed species information access

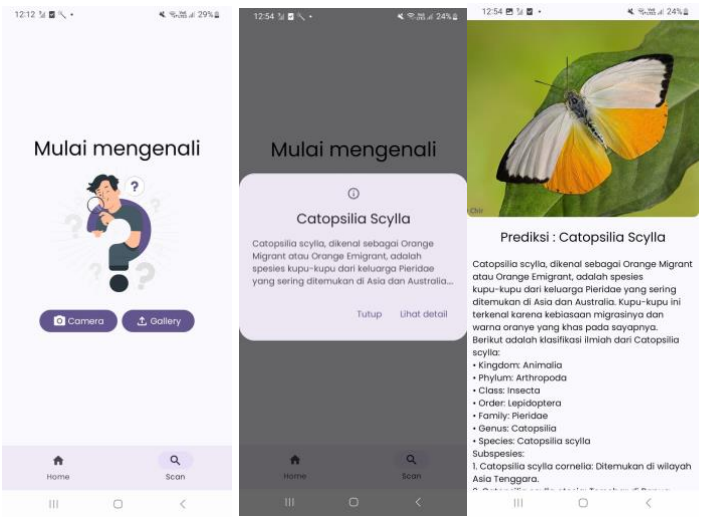

Fig. 12: Butterfly prediction page display design

4. Conclusion

This research demonstrates the successful implementation of a butterfly species classification system using MobileNet V3 Large architecture, achieving a remarkable 0.94 accuracy rate with an 80:20 data split ratio and performance metrics exceeding 0.90 across all evaluation parameters (precision, recall, and F1-scores) for six butterfly species. The integration of Google Cloud Platform (GCP) with the Android application effectively addressed mobile resource constraints by externally managing the machine learning model, while the implementation of user-friendly features, particularly the butterfly library and scanning functionality, received positive feedback from primary school student respondents. These results validate the effectiveness of combining CNN-based classification with cloud computing for mobile species identification applications, establishing a robust framework that balances classification accuracy, resource efficiency, and user engagement.

Acknowledgement

The authors would like to express their gratitude to everyone who provided valuable feedback and support during the completion of this work.

References

[1] B. Falakhi, E. F. Achmal, M. Rizaldi, R. R. R. Athallah, and N. Yudistira, "Perbandingan Model AlexNet dan ResNet dalam Klasifikasi Citra Bunga Memanfaatkan Transfer Learning," Jurnal Ilmu Komputer dan Agri-Informatika, vol. 9, no. 1, pp. 70-78, 2022.
[2] E. Barus, K. M. Pardede, and J. A. Putri Br. Manjorang, "Transformasi Digital: Teknologi Cloud Computing dalam Efisiensi Akuntansi," Jurnal Sains dan Teknologi, vol. 5, no. 3, pp. 904-911, 2024.
[3] F. B. Prasetio and T. Wellem, "Perancangan Dan Implementasi Aplikasi Android Untuk Layanan Informasi Pariwisata," IT-Explore: Jurnal Penerapan Teknologi Informasi dan Komunikasi, vol. 1, no. 2, pp. 114-132, 2022.
[4] F. F. Maulana and N. Rochmawati, "Klasifikasi Citra Buah Menggunakan Convolutional Neural Network," Journal of Informatics and Computer Science, vol. 1, no. 2, pp. 104-108, 2020.
[5] G. Ashari Rakhmat and M. Fikri Haekal, "Peningkatan Performa MobilenetV3 dengan Squeeze-and-Excitation (Studi Kasus Klasifikasi Kesegaran Ikan Berdasarkan Mata Ikan)," Journal MIND Journal, vol. 8, no. 1, pp. 27-41, 2023.
[6] I. Mahendra and D. T. Eby Yanto, "Sistem Informasi Pengajuan Kredit Berbasis Web Menggunakan Agile Development Methods Pada Bank Bri Unit Kolonel Sugiono," Jurnal Teknologi Dan Open Source, vol. 1, no. 2, pp. 13-24, 2018.
[7] K. Azmi and S. Defit, "Implementasi Convolutional Neural Network (CNN) Untuk Klasifikasi Batik Tanah Liat Sumatera Barat," vol. 16, no. 1, pp. 2580-2582, 2023.
[8] M. M. Lucini, P. J. Van Leeuwen, and M. Pulido, "Model error estimation using the expectation maximization algorithm and a particle flow filter," SIAM-ASA Journal on Uncertainty Quantification, vol. 9, no. 2, pp. 681-707, 2021.
[9] M. Romzi and B. Kurniawan, "Pembelajaran Pemrograman Python Dengan Pendekatan Logika Algoritma," JTIM: Jurnal Teknik Informatika Mahakarya, vol. 3, no. 2, pp. 37-44, 2020.
[10] A. Febriandirza, "Perancangan Aplikasi Absensi Online Dengan Menggunakan Bahasa Pemrograman Kotlin," Pseudocode, vol. 6, no. 1, pp. 53-59, 2019.
[11] A. D. Nugroho and W. M. Baihaqi, "Improved YOLOv5 with Backbone Replacement to MobileNet V3s for School Attribute Detection," SinkrOn, vol. 8, no. 3, pp. 1944-1954, 2023.
[12] A. Mujahid, M. Y. Abdullah, S. Suharya, and A. R. Adriansyah, "Analisis dan Pengembangan Sistem Informasi Pengelolaan Masjid berbasis Mobile dengan Teknologi API Web Service," Jurnal Informatika Terpadu, vol. 7, no. 2, pp. 80-86, 2021.
[13] P. Fernando, I. Junaedi, and A. Budi Yulianto, "Perancangan Sistem Informasi Booking Studio Musik Berbasis Website Di Studio Abe Music Dengan Metode Waterfall," Jurnal Sains dan Teknologi Widyaloka, vol. 2, no. 2, pp. 179-205, 2023.
[14] P. Palupiningsih, A. R. Sujiwanto, and R. R. B. P. Prawirodirjo, "Analisis Perbandingan Performa Model Klasifikasi Kesehatan Daun Tomat menggunakan arsitektur VGG, MobileNet, dan Inception V3," Jurnal Ilmu Komputer dan Agri-Informatika, vol. 10, no. 1, pp. 98-110, 2023.
[15] Y. Miftahuddin and F. Adani, "Sistem Klasifikasi Jenis Kupu-Kupu Menggunakan Visual Geometry Group 16," vol. X, no. X, pp. 1-11, 2022.