Jurnal Teknologi Informasi dan Pendidikan
Volume 18, No. 2, September 2025
https://doi.org/10.24036/jtip.v18i2.969

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

Real-Time Color Classification of Objects with an Improved MobileNetV2 CNN Model

Resti Apriliyanti[1], Denny Kurniadi[1*✉], Dony Novaliendry[1], Sandi Rahmadika[1], Muhammad Farhan[1]

[1]Faculty of Engineering, Universitas Negeri Padang, Padang, Indonesia

**Corresponding Author: dennykurniadi@ft.unp.ac.id*

Article Information

Article history:

No. 969
Rec. May 30, 2025
Rev. July 16, 2025
Acc. July 21, 2025
Pub. July 23, 2025
Page. 318 – 333

Keywords:

- *Color classification*
- *Convolutional Neural Network*
- *MobileNetV2*
- *Transfer learning*
- *Graphical User Interface*
- *Real-time system*

ABSTRACT

This research aimed to develop a Convolutional Neural Network (CNN) model for automatic object color classification using MobileNetV2. To determine the optimal configuration, the training process adjusted several hyperparameters, with particular focus on identifying the most suitable learning rate. The dataset consisted of 3,212 images grouped into five color categories: red, green, blue, random (including yellow, orange, and brown), and none (no object detected). Data augmentation techniques were applied to enhance the variety and robustness of the dataset. The model was trained using the Adam optimizer alongside the categorical crossentropy loss function, with various learning rate settings tested during training. Evaluation results showed that the model worked best with a learning rate of 0.0001 and a batch size of 32, with an average accuracy of 94%. To display prediction results in real time, the top-performing model was integrated into a graphical user interface (GUI). These findings demonstrate the effectiveness of the MobileNetV2-based CNN model in recognizing object colors and highlight its suitability for integration into real-time industrial sorting applications.

How to Cite:

Apriliyanti, R., & et al. (2025). Real-Time Object Color Classification Using an Optimized MobileNetV2 CNN Model. Jurnal Teknologi Informasi Dan Pendidikan, 18(2), 318-333. https://doi.org/10.24036/jtip.v18i2.969

This open-access article is distributed under the Creative Commons Attribution-ShareAlike 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. ©2023 by Jurnal Teknologi Informasi dan Pendidikan.

1. INTRODUCTION

Artificial intelligence (AI) has had a major impact on two industrial sectors, digital image processing and computer vision. A significant use of artificial intelligence involves categorizing objects based on their color, which serves as a fundamental component in automated systems for color-based sorting [1]. To achieve accurate sorting results, color-

based separation is essential for production efficiency [2], [3]. Despite technological advancements, many businesses still sort colors by hand. This method suffers from several drawbacks, including inconsistent results, operator fatigue affecting accuracy, and higher labor and time requirements [4].

Additionally, conventional color sensor-based systems have limitations in handling variations in lighting conditions and distinguishing similar color spectra [2], [5]. These systems are often not flexible because hardware needs to be modified when operating conditions change [6]. To overcome these challenges, deep learning techniques based on Convolutional Neural Networks (CNNs) offer promising solutions. CNNs are capable of automatically and efficiently extracting image features without requiring explicit feature engineering [5]. Moreover, they provide flexibility and do not necessitate hardware changes when input conditions vary.

In recent years, MobileNetV2 has emerged as one of the most popular CNN architectures for real-time applications due to its effectiveness and adaptability in feature extraction [7]. Rema (2023) found that MobileNetV2 achieved nearly equivalent accuracy to more advanced CNN models like ResNet50 [8] while using 70% fewer parameters and 40% less inference time. In the same way, Pamungkas & Suhendar (2024) confirmed the stability of MobileNetV2 under varying lighting conditions, making it suitable for software-based implementations without hardware dependencies [9].

Therefore, this study adopts the MobileNetV2-based CNN model as the core framework for color classification. The focus on RGB (Red, Green, Blue) colors was selected not only because of their visual distinctiveness but also because they form the basic representation in digital color models. This choice facilitates initial algorithm validation without compromising the model's ability to recognize other color patterns [10].

The objective of this research is to develop an accurate, efficient, and robust color classification system using the MobileNetV2 architecture. The model will be implemented into a Graphical User Interface (GUI) to enable real-time prediction visualization from webcam inputs, demonstrating its applicability in industrial sorting environments [11].

2. RESEARCH METHOD

A color identification system utilizing the MobileNetV2 convolutional neural network was developed in this study to classify objects captured via webcam into five distinct categories: red, green, blue, miscellaneous (yellow, orange, brown), and null (no object present). The overall workflow is illustrated in Figure 1.

Jurnal Teknologi Informasi dan Pendidikan
Volume 18, No. 2, September 2025
https://doi.org/10.24036/jtip.v18i2.969

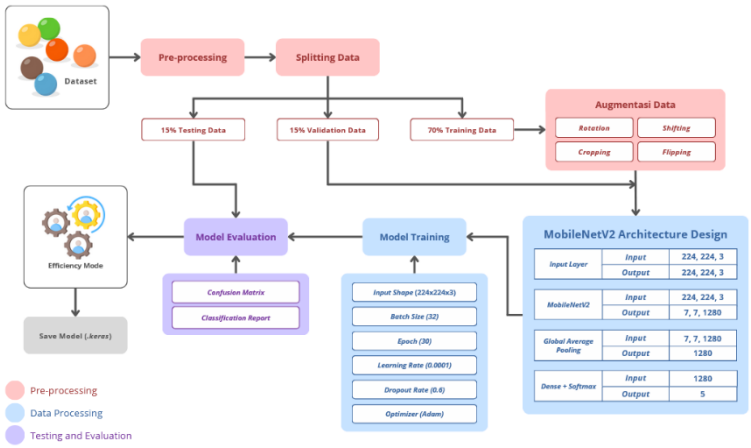


Figure 1. Workflow of the system modeling with MobileNetV2 architecture

2.1. Dataset Collection

The dataset was collected manually using a video-based capture method. Short video clips of colored objects were captured with a webcam to enable data gathering in a variety of visual circumstances. Following recording, every video was transformed into a separate JPG or JPEG image. This method was chosen because it allows for the rapid acquisition of large numbers of images from a single recording while also providing more natural variations in movement and perspective compared to static image capture.

The dataset consists of 3,212 colored object images categorized into five classes: red, green, blue, random (yellow, orange, brown), and none (no object detected). The dataset was split into training (70%), testing (15%), and validation (15%) sets, as shown in Table 1.

Table 1. Dataset distribution

Class	Training (70%)	Testing (15%)	Validation (15%)	Total
Red	453	97	98	648
Green	426	91	92	609
Blue	438	94	95	627
Random	506	108	110	724
None	422	90	92	604

In total, there are 2,245 training images, 480 testing images, and 487 validation images.

2.2. Preprocessing and Data Augmentation

Before the training phase began, all images were resized to a standard dimension of 224 × 224 × 3 pixels to align with the input requirements of the MobileNetV2 architecture. To enhance model convergence, pixel values were normalized from the original range of [0–

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

255] to [0–1]. Several methods of data augmentation were used to increase the size of the dataset and reduce the possibility of overfitting [4]. Figure 2 provides instances of these augmented images.

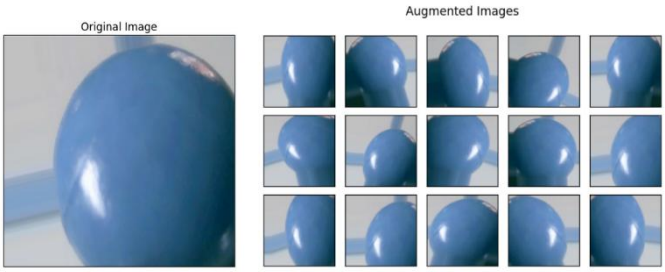


Figure 2. Sample results of data augmentation

Figure 2 illustrates the use of Keras' ImageDataGenerator module for generating additional training samples. The transformations applied included rotation (±20°), width and height shifting (±0.2), zooming (±0.2), shearing (±0.2), horizontal flipping, and rescaling (1./255).

2.3. MobileNetV2 Architecture Design

In contrast to traditional CNN architectures, MobileNetV2 uses depthwise separable convolutions, which are made up of two operations: depthwise and pointwise convolution layers [10]. Depthwise convolution applies filters independently to each input channel, effectively lowering computational complexity and the number of trainable parameters. Pointwise convolution then integrates these outputs using 1×1 filters to generate new feature representations. The model implemented in this research is built upon the pre-trained MobileNetV2 framework, adapted for object color classification. Figure 3 provides a graphic representation of the main steps of the model architecture.

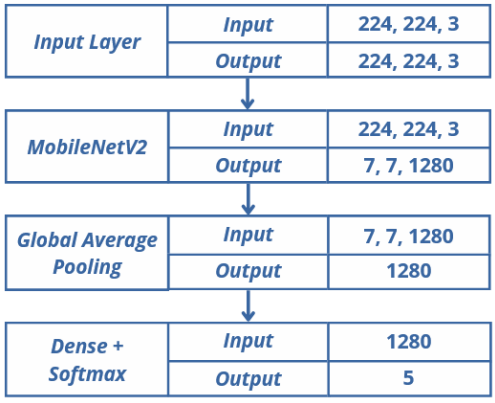


Figure 3. Architecture of the proposed model

Figure 3 shows the construction of the model schematically. Each layer plays a crucial role in transforming the input image into final color classification predictions. The function and purpose of each layer are explained in detail below.

2.3.1 Input Layer

The input layer receives images with a standard size of 224 × 224 pixels and three color channels (RGB). This format aligns with the input requirements of the MobileNetV2 architecture. The output from the input layer remains at 224 × 224 × 3, maintaining the original image dimensions and channel structure.

2.3.2 MobileNetV2

MobileNetV2 serves as the core of the model. It makes use of two essential components of depthwise separable convolutions. Depthwise convolution lowers the number of trainable parameters and the computational load by applying filters independently to each input channel. Then, pointwise convolution combines the outputs of depthwise convolution using 1×1 convolutions. High-level features taken from the input image are represented by the output, which is reduced to 7 × 7 × 1280 after going through several layers of MobileNetV2.

2.3.3 Global Average Pooling (GAP)

The Global Average Pooling (GAP) layer transforms the spatial output of MobileNetV2 into a compact 1280-dimensional feature vector by averaging values within each channel.

2.3.4 Dense + Softmax

A fully connected (dense) layer processes the 1280-dimensional feature vector to generate classification predictions. This assigns probability scores to each class label using the softmax activation function. In this instance, there are five classes: random (yellow, orange, brown), green, blue, red, and none (no object detected).

2.4. Model Training

The pre-trained model's underlying convolutional layers already possess useful feature extraction capabilities, hence only the classification head was taught during training, and all feature extraction layers were frozen [12]. A fixed batch size of 32 and a maximum of 30 epochs were used for training, and various learning rate combinations were used to find the best setup. Training was continued until improvements in accuracy and reductions in loss were observed without signs of overfitting

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

2.5. Model Evaluation

Model performance was evaluated using conventional classification metrics, including accuracy, precision, recall, and F1-score [10]. These are described as follows: the model's total accuracy measures how frequently samples are correctly classified. It is calculated by dividing the number of correct predictions (true positives and true negatives) by the total number of predictions made. The precision metric quantifies the percentage of actual positive predictions among all cases that the model predicts as positive. Recall, also known as sensitivity or true positive rate, measures the model's ability to detect all actual positive instances and shows how many of them were correctly identified. A balanced score that accounts for both false positives and false negatives is the F1-Score. It is the harmonic mean of recall and precision. It is particularly useful in cases where the dataset is imbalanced. These metrics provide information about the model's classification performance and ensure resilience and generalization for all five color classes: red, green, blue, random (yellow, orange, brown), and none (no object detected).

2.6. GUI Implementation

The best-performing model was implemented into a graphical user interface (GUI) built using CustomTkinter to visualize real-time predictions from webcam input. The GUI includes a camera preview window, prediction label display, and inference control buttons for ease of use in practical applications.

3. RESULTS AND DISCUSSION

In this research, the collected dataset undergoes preprocessing and is divided into three subsets: training, testing, and validation, with a proportion of 70%, 15%, and 15%, respectively. Increasing and improving data diversity through the use of data augmentation in the training set improves model generalization and reduces overfitting. The prepared data is then used to train the model over a range of scenarios with different learning rate values. The training process employs the Adam optimizer and uses categorical crossentropy as the loss function to guide the optimization of model parameters.

3.1. Model Baseline

To substantiate the practicality of the developed model, a comparative review was carried out between the MobileNetV2 architecture and conventional Convolutional Neural Networks (CNNs). Traditional CNN structures generally incorporate stacked convolutional and pooling layers, followed by fully connected layers to perform classification. Despite their effectiveness in image classification tasks, conventional deep learning models often demand large-scale datasets and extended training periods to reach optimal performance

Furthermore, when applied to limited datasets, classic CNNs often suffer from overfitting and exhibit weak adaptability to unseen data [10], [12].

Conversely, MobileNetV2 is engineered for efficiency, leveraging depthwise separable convolutions combined with inverted residual blocks and linear bottlenecks [7], [13]. This design drastically reduces both computational load and the total number of parameters, making it well-suited for deployment in environments with constrained resources. Furthermore, the model takes advantage of transfer learning by initializing its parameters with weights obtained from ImageNet-pretrained networks, enabling faster and more stable optimization throughout the training phase [14].

Empirical evaluations have consistently highlighted MobileNetV2's ability to deliver accurate and consistent results in image classification tasks, particularly under diverse lighting and background conditions [7], [9], [13]. Given its efficiency and robustness, its adoption in this research is considered appropriate, especially for real-time color recognition systems that rely on webcam input where both speed and accuracy are essential.

3.2. Learning Rate Evaluation

The MobileNetV2-based CNN model was trained using several learning rate values:0.1, 0.01, 0.001, 0.0001, and 0.00001. These values were selected based on their common usage in CNN training, particularly in transfer learning scenarios [7], [13]. The model was trained for a maximum of 30 epochs with a batch size of 32. This batch size was chosen as it provides a good balance between training speed, model stability, and generalization capability [15]. Additionally, batch size 32 is widely adopted in similar studies due to its ability to achieve high accuracy without overburdening computational resources [9], [10]. The variations in learning rate explored during the experiments are visualized in Figures 4.

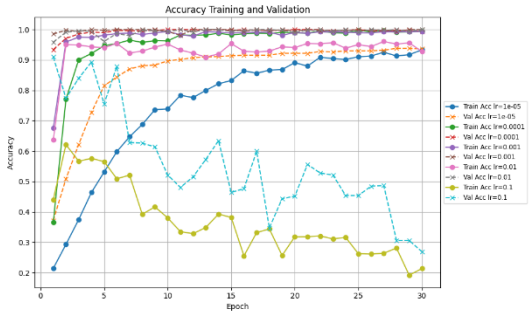


Figure 4. Training and validation accuracy across learning rates

Figure 4 shows how the learning rate employed has a major impact on the model's performance. High learning rates (0.1 and 0.01) caused unstable behavior, where the model failed to consistently improve its accuracy and showed signs of divergence, indicated by

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

fluctuating curves. In contrast, moderate learning rates (0.001 and 0.0001) led to stable accuracy improvements and achieved high performance (>95%) on both training and validation datasets. Meanwhile, the lowest learning rate (0.00001) resulted in slower accuracy improvement, although it still converged correctly toward the end of training.

Unpredictable performance with irregular variations in training and validation accuracy was the outcome of a learning rate of 0.1. Loss patterns did not regularly diminish, and validation accuracy ranged from 0.3 to 0.9, suggesting that this number was excessively high and impeded efficient learning. Learning rate 0.01 showed rapid increases in training accuracy but was not matched by equivalent improvements in validation accuracy. Validation accuracy stagnated around 0.91, suggesting early signs of overfitting due to an imbalance between training and generalization. Learning rate 0.001 achieved the fastest convergence, with training accuracy reaching 1.0 and validation accuracy stabilizing at approximately 0.98. This combination demonstrated excellent performance, efficient training, and strong generalization. Learning rate 0.0001 provided a fast increase in training accuracy up to around 0.98, with stable validation accuracy at approximately 0.92. This finding suggests that convergence speed and performance stability are well-balanced. A gradual but consistent performance improvement was demonstrated by learning rate 0.00001 (1e-5), which increased training accuracy from 0.2 to 0.92 and displayed a similar pattern in validation. Despite the comparatively high ultimate accuracy, this setting proved less effective for real-world training due to its delayed convergence.

Figure 4's training and validation accuracy curves offer valuable information about the model's classification performance, but it's also critical to examine the associated loss trends in order to evaluate the model's stability and learning behavior. Figure 5 presents the training and validation loss curves across different learning rate values.

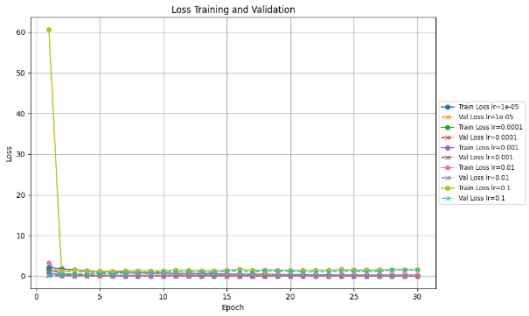


Figure 5. Training and validation loss across learning rates

Excessively high learning rates (0.1 and 0.01) hindered the model's ability to consistently lower loss levels, as seen in Figure 5. The loss curves for both training and validation showed oscillations and no discernible downward trend. On the other hand, moderate learning rates (0.001 and 0.0001) enabled rapid and stable loss reduction,

indicating that the model successfully learned image features. A smaller learning rate (0.00001) also reduced loss, albeit more slowly.

Learning rate 0.1 showed a sharp initial drop in loss from 60 to 1, followed by stagnation and fluctuation, especially in validation loss. With no discernible decrease in validation loss, training loss stabilized at 0.2 at a learning rate of 0.01. This trend pointed to early indicators of poor generalization and overfitting. Learning rate 0.001 achieved the most favorable results, with loss dropping below 0.1 and validation loss remaining highly stable. This configuration showed stability and speed.

Training loss rapidly decreased to about 0.1 at a learning rate of 0.0001, whereas validation loss remained relatively constant. This configuration reflected efficient training without indications of overfitting. In both training and validation, learning rate 0.00001 (1e-5) showed steady but sluggish loss reduction. Although steady, the convergence was noticeably sluggish. Table 2 provides a summary of the final evaluation results for various learning rates.

Table 2. Model performance across learning rates

Learning Rate	Accuracy (%)	Loss (%)
0.10000	26.90	1.50
0.01000	100.00	0.01
0.00100	100.00	0.00
0.00010	**99.38**	**0.03**
0.00001	93.63	0.22

Based on the experimental results, a learning rate of 0.0001 yielded the best overall performance in terms of stability and generalization. While 0.001 and 0.01 achieved 100% training accuracy, they were more prone to overfitting. Conversely, an excessively high learning rate (0.1) caused the model to fail in learning effectively, while an extremely low learning rate (0.00001) slowed down the training process despite producing acceptable results. The learning rate of 0.0001 provided high and stable accuracy between training and validation sets. The loss decreased sharply, then stabilized without any sign of overfitting. This configuration also achieved fast convergence (around epoch 5–10). The training accuracy and loss graphs for this optimal scheme are presented in Figure 6.

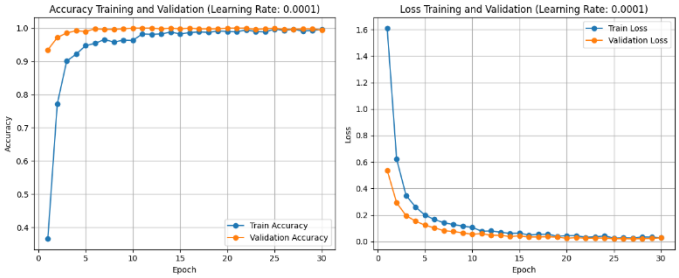


Figure 6. Best training scheme accuracy and loss curves

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

3.3. Confusion Matrix and Classification Report

In addition to overall accuracy and per-class metrics, a confusion matrix offers deeper insight into the model's behavior by showing how often each class is predicted correctly or confused with others. Figure 7 shows the resulting confusion matrix.

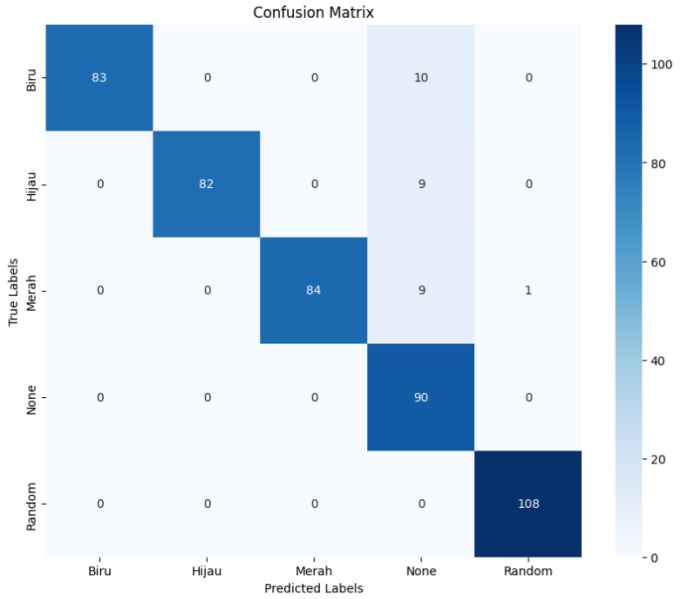


Figure 7. Confusion Matrix of the Best Model

Figure 7 displays the confusion matrix of the best-performing model, which was trained at a learning rate of 0.0001. The matrix shows that 84 red objects, 82 green objects, and 83 blue objects were accurately classified by the model. There were a few small misclassifications, though: nine green and red objects were mistakenly categorized as "none," and ten blue objects. These errors are insignificant compared to the total number of samples in each class. The model also demonstrated excellent performance on the none and random classes, achieving perfect classification accuracy for all 90 none samples and 108 random samples (which include yellow, orange, and brown objects). This indicates strong generalization capabilities across all five color categories. Overall, the confusion matrix confirms that the model achieves high classification accuracy across all color classes, with only minor misclassifications observed in the blue, green, and red categories. These results align with the evaluation metrics displayed in Table 3.

Table 3. Classification Report Metrics

	Precision	Recall	F1-Score
Blue	1.00	0.89	0.94
Green	1.00	0.90	0.95
Red	1.00	0.89	0.94
None	0.76	1.00	0.87

Volume 18, No. 2, September 2025
https://doi.org/10.24036/jtip.v18i2.969

Random	0.99	1.00	1.00
accuracy			0.94
macro avg	0.95	0.94	0.94
weighted avg	0.95	0.94	0.94

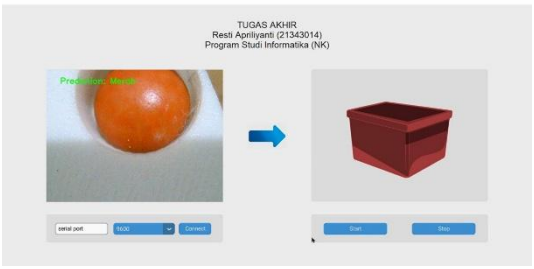
The model's 94% overall accuracy on the test dataset showed that it could distinguish between the five color classes. For the blue, green, and red classes, precision values reached 1.00, indicating that all positive predictions made by the model were correct. With recall values ranging from 0.89 to 0.90, the model was able to identify nearly all genuine positive cases among these groups. Precision and recall are fairly balanced in the respective F1-scores (~0.94–0.95).

The model obtained a precision of 0.99 and a flawless recall of 1.00 for the random class, which stands for yellow, orange, and brown. This indicates that no random-colored objects were overlooked during prediction, and nearly all predictions were correct. As seen in the confusion matrix, the model's significantly lower precision of 0.76 in the instance of the none class indicated some false positives. The recall score of 1.00, however, indicates that there were no false negatives and that all none-class samples were correctly detected. Even though the final F1-score of 0.87 is just behind that of other classes, it still represents respectable performance.

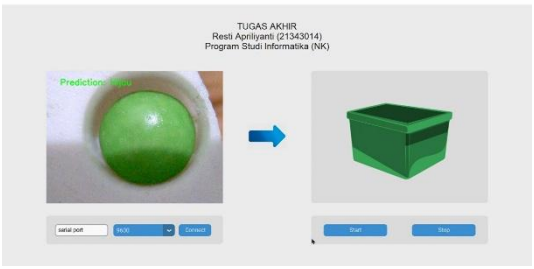
Together, these findings demonstrate that the CNN model based on MobileNetV2 exhibits good performance in all five color classes, with exceptional performance in recognizing objects of red, green, blue, and random colors. While the none class exhibits slightly reduced precision, the absence of false negatives ensures reliable detection of non-object frames, which is crucial for real-time industrial applications.

3.4. GUI Implementation

The best-performing model was integrated into a simple graphical user interface (GUI) made with CustomTkinter, a Python-based GUI tool. The CNN model may be deployed and validated in real-world scenarios thanks to the interface's ability to provide real-time predictions from camera input.

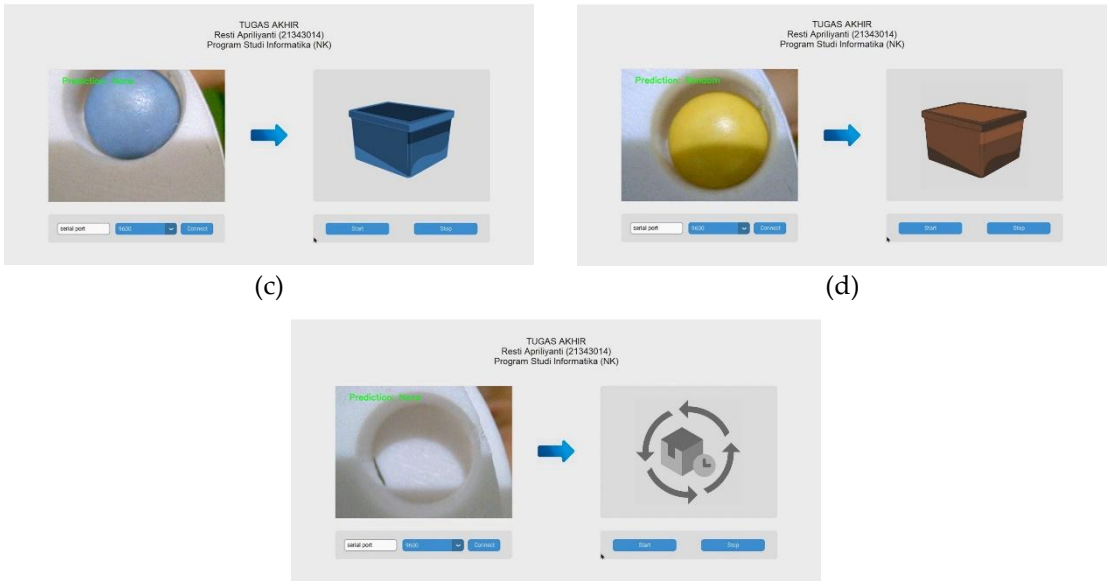


(a)



(b)

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id



(c) (d)

(e)

Figure 8. GUI for color classification (a) red, (b) green, (c) blue, (d) random, (e) none

Figures 8 illustrate the GUI developed in Python. Each figure represents the system output when detecting one of the five color classes: red, green, blue, random (yellow, orange, brown), and none (no object detected). These visualizations demonstrate how the system dynamically displays classification results based on real-time video input. The GUI was designed to facilitate monitoring during the color sorting process and ensure that the model could be effectively deployed in practical applications. Key features include dynamic display of predicted class labels, bounding box overlay corresponding to the detected color, and real-time webcam video stream.

This implementation supports both validation of the CNN model's accuracy and future integration with hardware components via an arduino microcontroller, making it suitable for industrial automation systems. The GUI operates by continuously capturing frames from the webcam and processing them through the trained MobileNetV2 model. Once an object is detected within the frame, the system predicts its color class and overlays a colored bounding box along with a label indicating the predicted class. This dynamic visualization enables users to monitor the system's performance in real time and verify prediction accuracy under varying environmental conditions.

Although the system performs well overall, there are some limitations in the current GUI implementation. One notable issue is prediction latency, which ranges between 300–700 ms , depending on the hardware specifications. Under testing conditions using a laptop equipped with an Intel Core i5 processor, 8 GB RAM, and a 1080p webcam, the system ran smoothly without significant delays.

Prediction accuracy is also impacted by camera stability and illumination quality. The system requires adequate lighting to allow the CNN model to detect color features optimally. In particular, glossy or reflective object surfaces can cause confusion between similar colors—especially between red and orange—leading to occasional misclassifications. Based on these results, the GUI is a useful tool for testing model performance; however, it would be more useful in industrial settings if it were faster and more resilient to difficult lighting circumstances.

3.5. Model Limitations

Despite the high classification performance of the MobileNetV2-based model, several misclassifications were observed during testing. The majority of these mistakes were made in difficult lighting situations, especially in places with extremely dim or overly bright lighting. Under such conditions, color features became less distinct, leading to confusion between similar color classes

This phenomenon is illustrated in Figure 13, which shows a comparison of red and orange colors under varying lighting conditions. The model had trouble telling these two colors apart in low light because of the decreased contrast. In overly bright conditions, color saturation caused red hues to appear more similar to orange tones, resulting in occasional misclassification of red objects as orange or even as part of the random class.



Figure 13. Comparison of red and orange colors under different lighting conditions

As shown in figure 13, in addition to lighting variations, visual noise such as shadows or reflections on object surfaces also affected prediction accuracy. These external elements created uncertainty in the feature extraction procedure and changed the apparent color qualities. This suggests that the suggested CNN model's sensitivity to lighting conditions is still a significant obstacle to practical use, even with its excellent generalization abilities. Preprocessing methods, adaptive normalization, or data augmentation under various illumination conditions could be used to overcome this constraint and enhance robustness and performance consistency across various settings.

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

3.6. Comparison with Previous Studies

The results of this study were compared with several previous studies that also applied CNN-based models for object or color classification tasks. Pamungkas & Suhendar [9], for instance, utilized a standard CNN architecture to classify plant leaf diseases and achieved an accuracy of approximately 91%. While their model demonstrated good performance, it required longer training time and showed limitations under varying lighting conditions. In contrast, the model developed in this research, which leverages MobileNetV2 with transfer learning and optimized hyperparameters, achieved a higher overall accuracy of 94% and exhibited strong robustness even when deployed in real-time GUI conditions.

Similarly, Rema [14] reported that MobileNetV2 could match the performance of deeper architectures like ResNet50 while maintaining computational efficiency. This aligns with the findings in this study, where MobileNetV2 was able to deliver high classification performance with a relatively low number of parameters, making it suitable for real-time applications. The model used by Allaam & Wibowo [10] for orchid genus classification faced challenges in generalization due to dataset limitations, which were mitigated in this research through effective data augmentation techniques.

Compared to the study by Ungkawa & Al Hakim [12] that used the Inception V3 architecture for fruit ripeness classification, our approach required less computational power while still maintaining comparable precision and recall values across multiple categories. Overall, the proposed model not only met but exceeded the performance metrics of similar studies, especially in terms of stability, convergence speed, and real-time applicability.

4. CONCLUSION

The experimental results indicate that the CNN model achieved optimal performance with a learning rate of 0.0001, reaching an average accuracy of 94%. The model achieved its highest performance using a learning rate of 0.0001 and a batch size of 32, making it the most reliable setup for this task. The model showed robust generalization, supported by consistently high precision, recall, and F1-score values exceeding 90% across four color categories: red, green, blue, and mixed tones (e.g., yellow, orange, brown).

The deployment of the model into a Graphical User Interface (GUI) demonstrated its ability to process and display live predictions from webcam input. The interface was developed with usability in mind, offering a clean and intuitive layout for real-time visualization of classification outputs and verification of detection reliability under actual conditions. Looking ahead, this color recognition system can serve as the foundation for an automated sorting mechanism when paired with hardware components like

microcontrollers and servo motors. Such integration could support real-time object separation in both industrial production lines and household automation systems.

These outcomes highlight the effectiveness of the MobileNetV2-based CNN architecture in performing accurate color identification and affirm its suitability for time-sensitive applications where rapid processing, efficiency, and consistent performance are essential requirements. These comparisons confirm the reliability and competitiveness of the proposed model for real-time object color classification tasks.

ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the academic advisors for their valuable feedback and guidance in improving this manuscript. Appreciation is also extended to all parties who contributed to the design, development, and implementation of the system throughout the research process.

REFERENCES

[1] K. Kisno, N. Fatmawati, R. Rizqiyani, S. Kurniasih, and E. M. Ratnasari, "Pemanfaatan Teknologi Artificial Intelligences (AI) Sebagai Respon Positif Mahasiswa Piaud dalam Kreativitas Pembelajaran dan Transformasi Digital," *IJIGAEd: Indonesian Journal of Islamic Golden Age Education*, vol. 4, no. 1, p. 44, Dec. 2023, doi: 10.32332/ijigaed.v4i1.7878.

[2] M. Firmansyah and P. Jaya, "Rancang Bangun Alat Pemilah Kematangan Buah Jeruk Manis Menggunakan Metode SVM," *Voteteknika (Vocational Teknik Elektronika dan Informatika)*, vol. 11, no. 4, p. 391, Dec. 2023, doi: 10.24036/voteteknika.v11i4.124547.

[3] M. Z. Andrekha and Y. Huda, "Deteksi Warna Manggis Menggunakan Pengolahan Citra dengan Opencv Python," *Voteteknika (Vocational Teknik Elektronika dan Informatika)*, vol. 9, no. 4, p. 27, Dec. 2021, doi: 10.24036/voteteknika.v9i4.114251.

[4] F. Nurdiyansyah *et al.*, "Penerapan Convolutional Neural Network Untuk Deteksi Kualitas Telur Ayam Ras Berdasarkan Warna Cangkang," *Jurnal MNEMONIC*, vol. 7, no. 1, pp. 40–47, 2024.

[5] A. Wibowo, Poningsih, I. Parlina, Suhanda, and A. Wanto, "Rancang Bangun Mesin Sortir Buah Kelapa Sawit Berdasarkan Tingkat Kematangan Menggunakan Sensor Warna TCS3200 Berbasis Arduino Uno," *STORAGE – Jurnal Ilmiah Teknik dan Ilmu Komputer*, vol. 1, no. 2, pp. 9–15, May 2022, doi: 10.55123.

[6] A. Chairi and R. Mukhaiyar, "Sistem Kontrol Color Sorting Machine Dengan Pengolahan Citra Digital," *JTEIN: Jurnal Teknik Elektro Indonesia*, vol. 4, no. 1, pp. 387–396, Jul. 2023, doi: 10.24036/jtein.v4i1.393.

[7] Y. Hatur Puspita and A. Sabri, "Transfer Learning Model Pralatih MobileNetV2 dan DenseNet121 untuk Klasifikasi Tanaman Rempah," *Jurnal Ilmiah Komputasi*, vol. 23, no. 1, pp. 67–74, Mar. 2024, doi: 10.32409/jikstik.23.1.3502.

[8] Y. O. L. Rema, "Deteksi Plat Nomor Kendaraan Bermotor dengan Segmentasi Gambar," *Jurnal Saintek Lahan Kering*, vol. 2, no. 1, pp. 20–23, Jun. 2019, doi: 10.32938/slk.v2i1.794.

[9] N. B. Pamungkas and A. Suhendar, "Penerapan Metode Convolutional Neural Network pada Sistem Klasifikasi Penyakit Tanaman Apel berdasarkan Citra Daun," *Edumatic: Jurnal Pendidikan Informatika*, vol. 8, no. 2, pp. 675–684, Dec. 2024, doi: 10.29408/edumatic.v8i2.27958.

[10] M. R. R. Allaam and A. T. Wibowo, "Klasifikasi Genus Tanaman Anggrek Menggunakan Metode Convolutional Neural Network (CNN)," *e-Proceeding of Engineering*, vol. 8, no. 2, pp. 3147–3179, Apr. 2021.

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

[11] W. Nengsih, J. N. S. Juni Nurma Sari, C. Angresta, and H. F. Dwinas, "DeepSun: Klasifikasi Fase Cahaya Matahari Berdasarkan Warna Menggunakan CNN," *Jurnal Komputer Terapan*, vol. 9, no. 2, pp. 182–190, Dec. 2023, doi: 10.35143/jkt.v9i2.6182.

[12] U. Ungkawa and G. Al Hakim, "Klasifikasi Warna pada Kematangan Buah Kopi Kuning menggunakan Metode CNN Inception V3," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 11, no. 3, p. 731, Jul. 2023, doi: 10.26760/elkomika.v11i3.731.

[13] A. Nada Nafisa, E. Nia Devina Br Purba, F. Aulia Alfarisi Harahap, and N. Adawiyah Putri, "Implementasi Algoritma Convolutional Neural Network Arsitektur Model MobileNetV2 dalam Klasifikasi Penyakit Tumor Otak Glioma, Pituitary dan Meningioma," *Jurnal Teknologi Informasi, Komputer dan Aplikasinya (JTIKA)*, vol. 5, no. 1, pp. 53–61, Mar. 2023, [Online]. Available: http://jtika.if.unram.ac.id/index.php/JTIKA/

[14] A. Z. Hibatullah, M. F. Rahman, and A. P. Sari, "Pemanfaatan Metode Convolutional Neural Network (CNN) Dengan Arsitektur MobileNetV2 Untuk Penilaian Kelayakan Rumah," *ALINIER JURNAL*, vol. 5, no. 2, pp. 129–139, Nov. 2024, [Online]. Available: www.elektro.itn.ac.id

[15] S. M. Hawibowo and I. Muhimmah, "Aplikasi Pendeteksi Tingkat Kematangan Pepaya menggunakan Metode Convolutional Neural Network (CNN) Berbasis Android," 2024.