JURNAL INFOTEL
VOL. 18, NO. 1, FEBRUARY 2026, PP. 77–93. DOI:10.20895/INFOTEL.V18I1.1499

RESEARCH ARTICLE

Implementation of MobileNetV2 Transfer Learning for Chicken Egg Quality Classification Using Jetson Nano

Dita Novita Sari[1], Yoeyong Rahsel[2], Akhmad Jayadi [3], Panji Andhika Pratomo [4*], and Dwi Ely Kurniawan[5]

[1,2]Information Systems, Faculty of Technology and Computer Science, Institut Bakti Nusantara
[3,4]Informatics Management, Department of Information Technology, Politeknik Negeri Lampung
[5]Informatics Engineering, Department of Informatics Engineering, Politeknik Negeri Batam

*Corresponding email: panjiandhikap@polinela.ac.id

Received: December 12, 2025; Revised: April 14, 2026; Accepted: April 23, 2026.

Abstract: Eggs are an important source of animal protein and are widely consumed by the public. Quality problems such as cracked or broken eggs are still frequently encountered during distribution and storage. Egg quality sorting has been done largely manually, making it prone to human error, time-consuming, and inconsistent. This study aims to develop a deep learning-based egg quality classification with a transfer learning approach using the MobileNetV2 architecture that is efficient for devices with limited computing capacity. The research method involves acquiring egg image datasets (good and broken), preprocessing data with normalization and augmentation, designing a MobileNetV2 model, conducting two-stage training (feature extraction and fine-tuning) and evaluating model performance. Implementation was carried out both in the development environment and on a Jetson Nano edge computing device to test the real-time application. The results showed that fine-tuning training increased classification accuracy to 92% with an average precision, recall, and F1-score of 0.95. Evaluation of the confusion matrix demonstrated the model's ability to distinguish egg classes well, although there were still small errors in the classification of "good" eggs. Implementation in the Jetson Nano demonstrated relatively fast inference times (50 to 70 ms) with low resource consumption, demonstrating the system's applicability at both farm and small-to-medium scale distributions. This research successfully presented an accurate, lightweight, and practically implementable egg classification model as a first step towards automating the egg sorting process in the livestock industry.

Keywords: Deep Learning, Egg, MobileNetV2, Quality, Transfer Learning

 ISSN: 2085-3688; e-ISSN: 2460-0997

1 Introduction

Eggs are a food ingredient that plays a vital role in meeting people's nutritional needs [1,2]. Besides being a cheap and readily available source of animal protein, eggs also have high economic value, leading to a continued increase in demand for this commodity annually. However, the high consumption and distribution of eggs also pose challenges in maintaining product quality before reaching consumers. One of the main problems frequently encountered is cracked or broken eggs during collection, transportation, and storage. Broken eggs not only reduce their market value but also accelerate the spoilage process by facilitating the entry of microorganisms into the egg interior. Therefore, efforts to maintain egg quality by separating good eggs from broken ones are crucial, both on a farm and in the distribution industry. Currently, egg sorting based on their physical condition is still largely done manually by workers [3, 4]. This manual process certainly has several drawbacks. First, visual sorting is relatively time-consuming because the number of eggs to be processed is usually very large. Second, the sorting results are highly dependent on the worker's meticulousness, making them prone to error. Third, this monotonous work has the potential to reduce the consistency of the selection results, especially if done over a long period of time. With the advancement of technology, particularly in the fields of digital image processing and artificial intelligence, opportunities have emerged to automate the egg classification process, making it faster, more accurate, and more consistent than manual methods.

In recent years, various studies have been conducted to utilize image processing technology to classify egg quality. Traditional methods commonly used include the extraction of color, texture, and shape features, which are then processed using simple classification algorithms such as k-nearest neighbor (k-NN) [5–7], support vector machine (SVM) [8, 9] and simple artificial neural networks. Although these methods can produce a fairly good level of accuracy, their weaknesses lie in their sensitivity to lighting conditions, object position, and image background. As a result, systems based on manual feature extraction often fail to maintain consistent performance when applied to diverse real-world conditions. With the development of deep learning methods, particularly convolutional neural networks, research in the field of object classification [10–12], including eggs, has experienced significant progress. CNNs are capable of automatically extracting features from image data without the need for manual feature design by researchers. Several studies have shown that CNNs can be used to classify various agricultural and livestock products, such as fruits, vegetables, and grains, with high accuracy. In the case of eggs, CNNs have been used to detect shell cracks and classify eggs based on external quality. However, most studies still utilize large and complex CNN architectures, requiring high-spec computing devices and consuming significant energy. This condition makes it difficult to implement on edge devices commonly used in the field, such as surveillance cameras or automation systems in small-scale farms.

One approach that is increasingly being used to address this problem is transfer learning. This technique utilizes a CNN model that has been previously trained on a large dataset, such as ImageNet, and then re-adapted to a specific dataset according to research needs. With transfer learning, researchers do not need to train the model from scratch, thus speeding up the training process and reducing the need for a smaller dataset. One popular architecture specifically designed for resource-constrained devices is MobileNetV2 [13–15]. This model has a relatively small number of parameters but is still capable of delivering

high accuracy on various image classification tasks. In this study, the authors utilized a transfer learning approach using MobileNetV2 to classify eggs into two classes: good eggs and broken eggs. The dataset used consisted of egg images obtained under various conditions with varying lighting and backgrounds. This study was conducted in two training stages. The first stage was carried out by training only the top layer (fully connected layer) while the main layer of MobileNetV2 was frozen as a feature extractor. The second stage was fine-tuning, which involved unlocking most of the MobileNetV2 layers to adjust the weights to the egg dataset. This strategy is expected to produce a more accurate model in recognizing specific characteristics of eggs. In addition to evaluating model performance based on accuracy, loss, precision, recall, and F1-score metrics, this study also included visual analysis using learning curves, confusion matrices, and ROC curves. This analysis is crucial for understanding the model's ability to distinguish between good and broken eggs, as well as identifying model weaknesses, such as misclassification of eggs with subtle, difficult-to-recognize cracks.

Another contribution of this study is the implementation of the model on an edge computing device, the Jetson Nano. This device is relatively inexpensive and energy-efficient, making it suitable for use on farms or small-scale industries. By conducting trials on the Jetson Nano, this study aims to demonstrate that a deep learning-based egg classification system is not only conceptual but can also be implemented in the field. The evaluation includes inference time, resource consumption, and system performance when run in real time. The results of these tests are expected to form the basis for the development of automated egg classification systems in industry. This research contributes to three main aspects. First, this study presents an efficient yet accurate transfer learning-based egg classification system with MobileNetV2. Second, this study compares the model's performance in two training stages, one without fine-tuning and one with fine-tuning, to demonstrate significant performance differences. Third, this study evaluates the model's implementation on the Jetson Nano device as a concrete representation of the application of deep learning technology in the livestock sector. It is hoped that this research can be the first step towards faster, more accurate, and more practical egg sorting automation, thereby improving product quality and operational efficiency in the egg distribution chain.

2 Research Method

This research was conducted with an experimental approach using a deep learning-based classification method. The research design consists of several main stages, namely (1) data acquisition, (2) data preprocessing, (3) classification model design with the MobileNetV2 architecture, (4) model training and fine-tuning, and (5) model performance evaluation both in the development environment (Google Colab) and on the Jetson Nano edge computing device. The research flowchart is shown in Figure 1.

2.1 Data Acquisition

The dataset used in this study consists of chicken egg images divided into two categories: good eggs and broken eggs. Data were obtained through direct documentation using a standard-resolution digital camera with artificial lighting to minimize shadows. The total dataset consists of 200 images with a balanced distribution between the two classes.

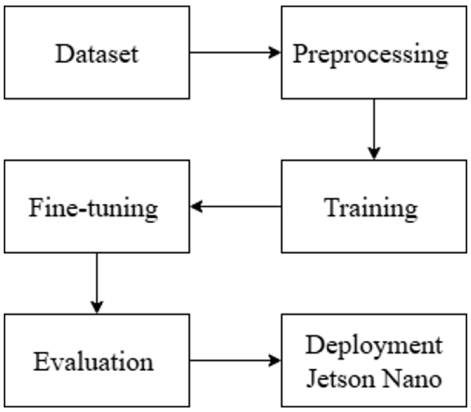

Figure 1: Research flow diagram.

The dataset was then divided into two subsets: a train set (80%) and a validation set (20%). This separation follows common practice in deep learning research to allow the model to learn on a large portion of the data while simultaneously testing its performance on previously unseen data [16–18]. The dataset structure is organized into separate directories to facilitate the retrieval process using the TensorFlow image generator. Each class (good and broken) is placed in a separate folder. The dataset is read using the `image_dataset_from_directory` function, which automatically assigns labels according to the folder name, resulting in consistent class labels without the need for additional manual annotation.

2.2 Data Preprocessing

Before being used in training, egg images undergo a preprocessing stage. First, the images are resized to 160×160 pixels to adapt to the MobileNetV2 architecture. Second, the images are normalized by dividing the pixel values from 0–255 into the range 0–1 using a Rescaling (1/255) layer. This normalization aims to speed up the training process and prevent scale differences between features. In addition, data augmentation techniques are also applied to expand the variety of training data, including rotation, translation, zooming, and horizontal flipping. Thus, the model can be more robust to variations in the position and lighting conditions of the eggs.

2.3 Transfer Learning

Transfer learning refers to a deep learning approach that utilizes pretrained weights from large-scale models, such as ImageNet, and applies them to a different dataset. This approach accelerates the training process, improves model performance, and helps minimize the risk of overfitting. It is widely applied in image classification tasks, including plant disease detection, as it can achieve optimal performance even with limited datasets [19]. The MobileNet architecture has a relatively smaller number of parameters compared to other convolutional neural networks (CNNs), making it suitable for deployment on resource-

constrained devices. The number of parameters of several popular transfer learning models is presented in Table 1.

In addition to the choice of architecture, the selection of training hyperparameters such as learning rate and batch size plays a critical role in determining model performance. The learning rate controls the update magnitude of model weights, where a higher value accelerates convergence but risks instability, while a lower value ensures more stable optimization, especially during fine-tuning; therefore, this study applies a higher learning rate in the feature extraction phase and a lower one during fine-tuning. Meanwhile, batch size affects both computational efficiency and training stability, and a moderate value is chosen to balance memory limitations on edge devices with stable gradient updates. Overall, proper tuning of these parameters is essential for achieving optimal performance, particularly when working with limited datasets [20].

Table 1: All parameter of other popular transfer learning models [21]

Model	**Number of Parameters**
GoogleNet	6.8 Million
AlexNet	60 Million
SqueezeNet	1.25 Million
VGG16	138 Million
MobileNet	4.2 Million

2.4 Model Design

The architecture used in this study is MobileNetV2, a lightweight CNN developed specifically for devices with limited computing. MobileNetV2 utilizes depth wise separable convolution and an inverted residual structure, significantly reducing the number of parameters without drastically reducing accuracy. The model was built using two training stages. In the first stage, weights from MobileNetV2, pre-trained on the ImageNet dataset, were used as feature extractors. The main layers of MobileNetV2 were frozen so that they did not undergo weight updates. Additional layers, including Global Average Pooling, Dropout, and a Dense layer with a sigmoid activation function, were added to generate outputs for good or broken eggs. The second stage was fine-tuning. In this stage, most of the MobileNetV2 layers were unfrozen so that the weights could be updated based on the egg dataset. Fine-tuning ensured that the model not only relied on general features from ImageNet but also adapted to specific characteristics of egg images, such as crack patterns on the shell.

2.5 Research Procedures

The general research procedure can be written in the form of an algorithm as follows:

1. Start
2. Get a dataset of egg images (both good and broken)
3. Split the dataset: 80% for training, 20% for validation
4. Apply preprocessing: resizing (160×160), normalization, and data augmentation

5. Load the MobileNetV2 architecture with pre-trained weights from ImageNet
6. Add a binary classification layer (Dense with sigmoid)
7. Train the model in stage 1: feature extraction for 10 epochs
8. Continue with stage 2: fine-tuning for 5 epochs
9. Save the trained model in `.keras` and `.tflite` formats
10. Test the model on the validation data and analyze the results (accuracy, loss, confusion matrix)
11. Deploy the model on the Jetson Nano for real-time testing
12. Finish

2.6 Evaluation and Testing

The evaluation of the performance of the model was conducted in two aspects: quantitative evaluation and implementation in the real-world. The quantitative evaluation included calculating metrics such as accuracy, precision, recall, and F1-score, as well as analyzing the confusion matrix. Furthermore, learning curves (accuracy and loss in training and validation data) were used to analyze overfitting and underfitting. Evaluations were also conducted on two training scenarios, before and after fine-tuning, to compare the impact of these strategies on model performance. For real-world implementation, the trained model was converted to TensorFlow Lite (.tflite) format to run on a Jetson Nano. The testing was conducted by providing new egg images, both from the test dataset and from a real-time camera, and then recording the inference time and prediction accuracy. These test results serve as an indicator of the usability of the model in field applications, particularly in small- and medium-scale industries.

2.7 Deployment

2.7.1 System Architecture Design

The block architecture of the system to be created is explained in Figure 2

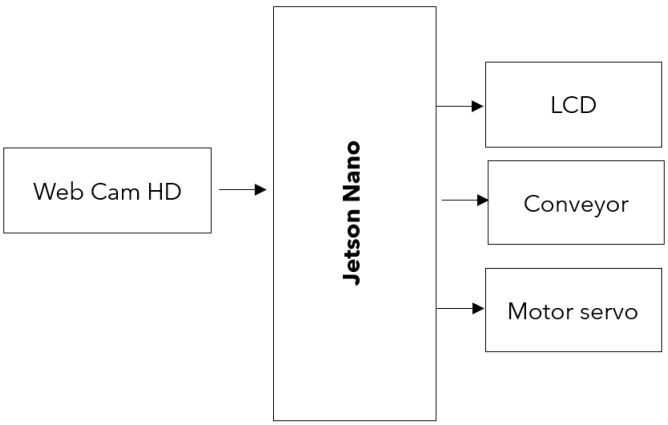

Figure 2: System block diagram architecture.

The block diagram depicts an embedded AI-based system that works through integrated input, process, and output stages. At the input stage, the HD Web Cam functions as an image acquisition device that captures real-time images or videos of eggs on the production line. This visual data is then sent to the Jetson Nano as the main processing unit, where computer vision and deep learning-based analysis is performed to identify egg characteristics, such as physical condition or quality category. At the output stage, the processing results are displayed via LCD as a medium for monitoring system information, as well as used as a basis for actuator control. The Jetson Nano sends control signals to the conveyor system to regulate egg movement and to the servo motor to carry out the sorting or transfer mechanism for eggs according to the classification results.

2.7.2 Implementation

In the implementation phase, the designed system will be realized in a tangible form by integrating hardware and software according to specification set out in Figure 3

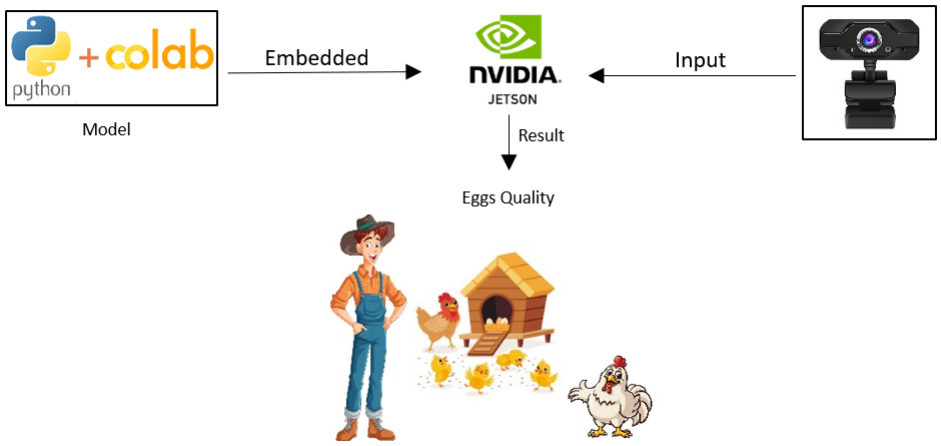

Figure 3: Implementation diagram.

The system implementation in the figure 3 shows the workflow of an embedded intelligent system for computer vision-based egg quality classification. At the embedded model stage, a deep learning model is developed and trained using Python in the Google Colab environment, then deployed to an NVIDIA Jetson device as an edge computing platform. At the processing stage, the camera serves as a source of real-time egg image input, which is then processed by Jetson using the embedded model to extract visual features and classify egg quality based on parameters learned during training. At the result stage, the system produces egg quality category information that can be used as a basis for decision-making in farm management, such as egg sorting, productivity monitoring, and livestock condition evaluation, thus supporting automation and increasing operational efficiency in a smart farming environment.

3 Results

3.1 Model Training Results

The model training process was conducted in two distinct stages: 50 epochs of training before fine-tuning (feature extraction) and 50 epochs of training after fine-tuning. In the initial training phase, with 50 epochs, the graph shows a steady increase in training accuracy, reaching over 90%. This indicates that the model has successfully learned the patterns from the training data. However, a different pattern is observed in validation accuracy, which tends to stagnate in the 50–70% range and fluctuates between epochs. This phenomenon indicates the model's limited ability to generalize to previously unseen data, resulting in the model not being optimal in distinguishing between "good" and "broken"eggs in the validation data. A situation where training accuracy continues to increase while validation accuracy remains relatively unchanged can be interpreted as an indication of underfitting in the model's feature representation. This is normal because, in the initial phase, training is only performed on the top layer of the transfer learning architecture without fine-tuning the base model weights. Thus, the features used for classification are still general and unable to capture the specific characteristics of the egg dataset used. Fluctuations in validation accuracy also indicate the model's sensitivity to variations in the validation data. This confirms the finding that in the initial stages, although the model has high training accuracy, its generalization ability is still limited. Therefore, additional fine-tuning strategies are needed to optimize the model's use of lower-layer weights to improve validation performance, as seen in Figure 4

Figure 4: Graph of training results before fine-tuning.

After fine-tuning by adding 50 training epochs, the graph shows significant improvements in both accuracy and loss. Accuracy on the training data increased rapidly, approaching 100%, while validation accuracy also consistently increased, reaching the 85–95% range. Furthermore, the previously sharp fluctuations in validation accuracy appeared to have decreased, indicating stable model performance on data not used during training. In terms of loss, both training and validation loss experienced a more stable de-

crease compared to the initial training, as seen in Figure 2. This demonstrates that the model not only successfully adjusted its weights to the training data but also maintained good performance on the validation data. Figure 5 shows the difference between the results before and after fine-tuning, highlighting the importance of adjusting the base model weights in transfer learning scenarios.

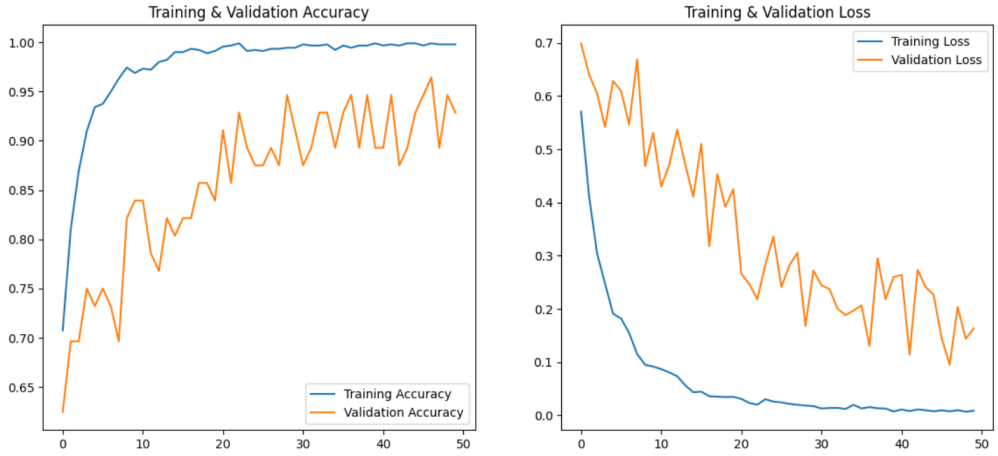

Figure 5: Graph of training results after fine-tuning.

In the initial stage, the base model weights are still general, derived from training on large datasets such as ImageNet. After fine-tuning, these layers become more adaptive to the specific characteristics of the egg dataset used. Therefore, the improved validation performance indicates that the model is able to capture more relevant visual patterns in distinguishing between "good" and "broken" eggs. This finding aligns with previous studies that reported that fine-tuning the pre-trained model architecture can improve accuracy on classification tasks with data domains different from the initial training data [22]. Therefore, it can be concluded that the use of transfer learning with a fine-tuning stage is an effective strategy in improving egg image classification performance in this study. Overall, accuracy during the feature extraction stage increased gradually, reaching an average of 85% on the validation data. After further fine-tuning, accuracy increased even further, reaching over 92%. The loss graph also showed a consistent decrease, indicating that the model successfully learned patterns from the data without significant signs of overfitting.

3.2 Model Performance Evaluation

Evaluation of model performance using validation data yields metrics as shown in Table 2.

Based on the classification report results in Table 2, evaluation scores indicate quite good model performance. For the "good" class, the model achieved a precision of 1.00, meaning all predictions categorized as "good" accurately matched the original class without error. However, a recall value of 0.91 indicates that approximately 9% of "good" data was still misclassified as "broken". Meanwhile, for the "broken" class, a recall of 1.00 was achieved, meaning all "broken" data was correctly classified by the model without any misses. How-

Table 2: Model evaluation results

	Precision	Recall	F1-Score	Support
Good	1.00	0.91	0.95	32
Broken	0.89	1.00	0.94	24
Accuracy			0.95	56
Macro Avg	0.94	0.95	0.95	56
Weighted Avg	0.95	0.95	0.95	56

ever, a precision value of 0.89 indicates that a small portion of "broken" predictions actually came from the "good" class.

Overall, the model achieved 92% accuracy, with an F1-score of 0.95 for both the "good" class and 94% for the "broken" class. The macro average and weighted average values were also both 0.95, indicating consistent model performance across both classes. These findings demonstrate that the model not only has high accuracy performance but is also quite balanced in classifying both classes, although there are still small errors in the "good" class. These results indicate that the transfer learning and fine-tuning approaches used are effective in improving classification quality, in line with previous studies reporting the effectiveness of similar methods on image classification problems [22–25, 25, 26].

3.3 Model Performance Comparasion

This study uses MobileNetV2 as the main model because it is known as a lightweight architecture with quite good performance on devices with limited resources. Technically, MobileNetV2 uses a depthwise separable convolution approach with stable ReLU6 activation, but has limitations in representing more complex features, especially on small datasets with subtle feature variations such as egg cracks. In addition, MobileNetV3 was chosen as a comparison in this model because this model is designed to improve the efficiency of feature representation while maintaining low computational complexity, making it more adaptive to the needs of real-time systems based on edge computing. Table 3 shows the comparative performance of the two models on the Jatson Nano device.

Table 3: Comparative performance

	MobileNetV2	MobileNetV2
Akurasi	92%	94%
F1-Score (Good)	0.95	0.97
F1-Score (Broken)	0.94	0.96
CPU Load	40%	45%
GPU Load	70%	60%
RAM Usage	550 MB	600 MB

3.4 Confusion Matrix Analysis

Based on the confusion matrix results in Figure 6, it can be seen that the model performed quite well in distinguishing between the "good" and "broken" classes. Of the 32 "good"

egg samples, 29 were correctly classified, while 3 were incorrectly classified as “broken”. On the other hand, all 24 “broken” egg samples were correctly classified without error.

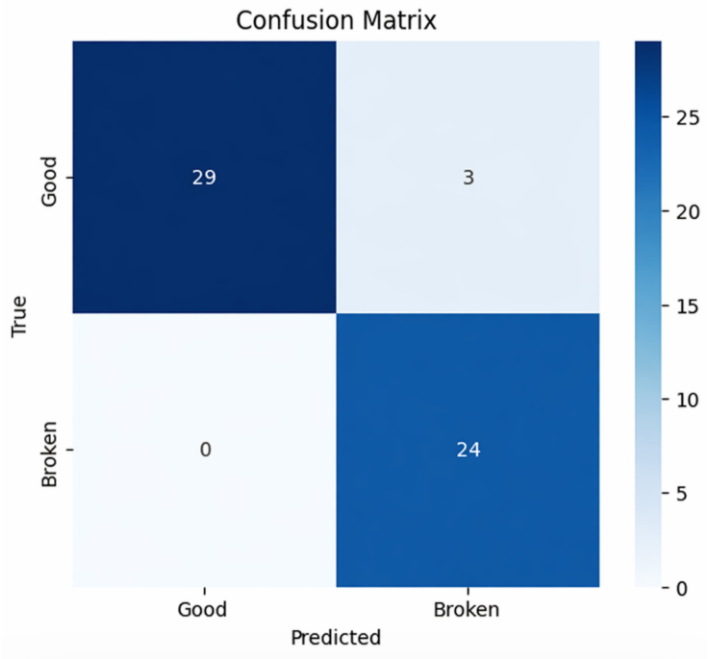

Figure 6: Confusion matrix.

These results indicate that the model has a high level of accuracy, particularly in the “broken” class, which achieved perfect classification (100% recall). However, there were small errors in the "good" class, which could be caused by certain visual similarities, such as lighting, shooting angle, or the egg’s surface condition resembling a crack. Thus, this confusion matrix confirms the previous findings from the accuracy and loss graphs, while also providing a more detailed picture of the model’s strengths and weaknesses in each class.

3.5 Hardware Implementation

Figure 7 shows the implementation of a deep learning-based egg classification system using an NVIDIA Jetson Nano device, where a controlled lighting environment is essential to minimize shadows and glare that may be misinterpreted by the model as cracks or defects. The use of diffuse ring lighting, positioned around the camera lens, provides uniform illumination across the egg surface and reduces shadow artifacts, while backlighting by placing a light source beneath the egg enhances the visibility of internal structures and shell integrity, enabling the model to better distinguish between superficial dirt and actual cracks. In addition, background contrast and color play a critical role in improving feature extraction; therefore, a matte black, non-reflective background is recommended to maximize the signal-to-noise ratio (SNR), ensuring that the model focuses on relevant egg features rather than background interference. Camera positioning and multi-view acquisition

further support system robustness, where the camera is mounted in a top-down orthogonal position (90°) to minimize perspective distortion, and a mechanical rotation system allows full 360° surface inspection.

Together with these factors, the conveyor speed was carefully controlled during laboratory experiments to ensure consistent image acquisition. A stable conveyor speed enables the camera system to capture images without motion blur and maintains synchronization between object positioning and inference timing. In contrast, variations in conveyor speed may introduce motion artifacts or inconsistent object framing, which can negatively impact classification accuracy and overall system reliability.

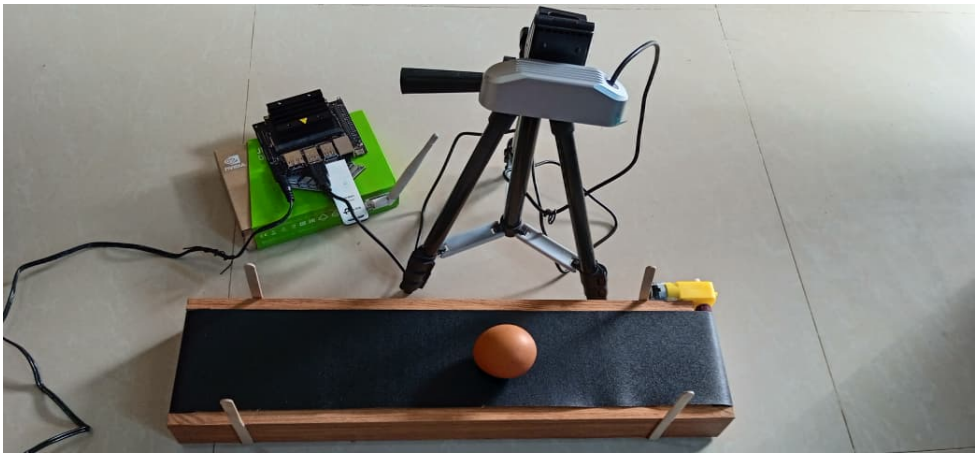

Figure 7: Hardware implementation.

Eggs were placed on a DC motor-driven conveyor track to simulate an automated sorting process. With this design, the system can detect and classify eggs into two categories: "good" and "broken." The resulting classification information can then be used to control the sorting actuators in the next stage. This implementation demonstrates that the model trained in Google Colab can be run on low-power devices like the Jetson Nano without significant performance loss. This demonstrates the system's potential for practical application in the livestock industry or egg distribution centers, where manual classification is still widely used.

4 Discussion

The comparative performance between MobileNetV2 and MobileNetV3, as presented in Table 1 , demonstrates notable differences in classification accuracy, computational efficiency, and resource utilization on the Jetson Nano device. MobileNetV2, which serves as the primary model in this study, achieves an accuracy of 92% with F1-scores of 0.95 for the "good" egg class and 0.94 for the "broken" egg class, indicating reliable classification performance under computational constraints. However, MobileNetV3 shows improved predictive capability, achieving a higher accuracy of 94% along with enhanced F1-scores of 0.97 and 0.96 for "good" and "broken" classes, respectively. This improvement reflects the

effectiveness of architectural enhancements in MobileNetV3, particularly in capturing subtle feature variations such as minor cracks on egg surfaces. From a computational perspective, MobileNetV3 exhibits a slightly higher CPU load (45%) compared to MobileNetV2 (40%), and a marginal increase in RAM usage (600 MB vs. 550 MB), indicating a modest increase in resource consumption. Nevertheless, MobileNetV3 demonstrates better GPU efficiency, with a lower GPU load of 60% compared to 70% in MobileNetV2, suggesting more optimized parallel computation during inference. Overall, these results indicate that while MobileNetV2 provides a balanced and efficient baseline for edge deployment compared to MobileNetV3.

4.1 Overfitting Analysis in the Initial Training Stage (Before Fine-Tuning)

Figure 4 displays the model's training and validation curves during the initial training phase, where all layers of the MobileNetV2 base model are still frozen (feature extraction). At this stage, significant overfitting is observed, indicated by a significant increase in training accuracy, while validation accuracy tends to stagnate and fluctuate. The complete data are shown in Table 4.

Table 4: Model overfitting indicators

Evaluation Indicators	**Training**	**Validation**	**Analysis**
Final Accuracy (Last Epoch)	0.90	0.60	A large difference in accuracy indicates the model is only learning on the training data
Average Accuracy	High and stable	Low and fluctuating	The model fails to generalize.
Final Loss	0.25	0.80	The validation loss is much higher → overfitting.
Accuracy Trend	Increasing consistently	Unstable	The model memorizes the training data patterns.
Loss Trend	Steady decline	Tends to increase	The model's inability to adapt to new data.
Training–Validation Accuracy Gap	30%	—	A large gap is a strong sign of overfitting.
Curve Stability	Stable	Very volatile	The validation dataset is not well-learned.

Table 4 summarizes the quantitative indicators that indicate overfitting in the initial training phase prior to fine-tuning. Although the accuracy of the training data consistently increased to nearly 90%, the performance of the validation data remained low and fluctuated. Furthermore, the high validation loss, which tended to increase inversely with the training loss, indicated that the model was too adapted to the training data and did not generalize. This confirms that the MobileNetV2 architecture used in the initial phase was relatively complex compared to the amount and diversity of training data. The overfitting that occurred was also influenced by the limited amount of training data and the complexity of the MobileNetV2 architecture, necessitating fine-tuning with a lower learning

rate and a more controlled number of epochs. Thus, the results in Figure 4 do not indicate model failure, but rather serve as a basis for consideration for implementing a further training stage through fine-tuning.

4.2 The Role of Fine-Tuning in Reducing Overfitting

Fine-tuning was applied to address the overfitting problem that occurred in the initial training stage, as shown in subsection 4.1. In this stage, all previously frozen MobileNetV2 Dense layers were unfrozen and retrained using a lower learning rate [27, 28]. This strategy allowed the model to fine-tune its weights to the characteristics of the data without compromising the representation of common features learned from the ImageNet dataset.

The comparison of overfitting indicators before and after fine-tuning, as shown in Table 5, reveals a significant change in the model's learning behavior. In the pre-fine-tuning stage, the model demonstrated high training accuracy, but not comparable performance on the validation data. This was reflected in the large gap between training and validation accuracy and the relatively high and increasing validation loss value, which are classic indicators of overfitting.

Table 5: Comparison of overfitting indicators before and after fine-tuning

Indicators	**Before Fine-Tuning**	**After Fine-Tuning**	**Analysis**
Training Accuracy	High $\approx 90\%$	High and stable	Consistency.
Validation Accuracy	Low and fluctuating	More stable and increasing	Improved generalization.
Training Loss	Sharply decreasing	Decreasing and stable	Not overoptimistic.
Validation Loss	High and increasing	Decreasing and stable	Reduced overfitting.
Accuracy Gap	Large	Smaller	Indicators of fine-tuning success.

After the fine-tuning process was implemented, clear improvements in these indicators were observed. The validation accuracy increased and became more stable, while the gap between training and the validation accuracy decreased significantly. The decrease and stabilization of the validation loss also indicated that the model was no longer overfitting the training data and was instead beginning to build a more general feature representation. Furthermore, although training accuracy remained high, the rate of decrease in training loss became more stable compared to the previous stage. This indicates that relearning in MobileNetV2 layers with a lower learning rate was able to control model complexity more effectively. Thus, fine-tuning functions as a weight adaptation mechanism that improves the balance between model capacity and the amount of available data. In general, the results in Table 5 confirm that the application of fine-tuning not only improved the quantitative performance of the model, but also directly reduced the level of overfitting that occurred in the initial training stage. These findings strengthen the argument that fine-tuning strategies are a critical component in ensuring the model can be reliably implemented in real-world environments, such as camera-based egg classification systems on embedded devices.

5 Conclusion

This study proposes and validates a lightweight deep learning-based system for egg quality classification using a transfer learning approach with the MobileNetV2 architecture. The experimental results demonstrate that the proposed model achieves a classification accuracy of 92–95% after fine-tuning, with precision, recall, and F1-score consistently above 0.92, indicating robust discriminative performance between good and broken egg classes. The two-stage training strategy (50 epochs for feature extraction and 50 epochs for fine-tuning) effectively improves feature specialization while maintaining generalization capability, particularly under limited dataset conditions. From a deployment perspective, the model was successfully optimized in TensorFlow Lite format and implemented on a Jetson Nano edge device, achieving an average inference time of 50–70 ms per image. This performance confirms that the proposed system satisfies real-time processing requirements while maintaining computational efficiency in resource-constrained environments.

Scientifically, this study contributes by demonstrating that a lightweight transfer learning framework can achieve a favorable trade-off between accuracy and latency in embedded smart farming applications. However, limitations remain in terms of dataset size and class diversity, which may affect model generalizability. Therefore, future work should focus on expanding the dataset without noise, incorporating more complex egg quality categories (e.g., micro-cracks and deformities) and validating the system across different environmental conditions to ensure robustness and scalability.

Acknowledgments

This research was funded by the Ministry of Higher Education, Science and Technology (107/LL2/DT.05.00/PL/2025) and Lampung State Polytechnic and Batam State Polytechnic for implementing this joint research.

References

[1] M. J. Puglisi and M. L. Fernandez, "The health benefits of egg protein," *Nutrients*, vol. 14, no. 14, p. 2904, 2022.

[2] J. Gautron, C. Dombre, F. Nau, C. Feidt, and L. Guillier, "Production factors affecting the quality of chicken table eggs and egg products in Europe," *Animal*, vol. 16, 2022.

[3] A. Nasiri, M. Omid, and A. Taheri-Garavand, "An automatic sorting system for unwashed eggs using deep learning," *Journal of Food Engineering*, vol. 283, p. 110036, 2020.

[4] B. P. Swastika, A. Rizal, H. Mukhtar, M. F. Bahrudin, *et al.*, "Low cost automatic egg sorting system using conveyor with light sensor," pp. 528–532, 2024.

[5] T. Hayit, A. Endes, and F. Hayit, "Knn-based approach for the classification of fusarium wilt disease in chickpea based on color and texture features," *European Journal of Plant Pathology*, vol. 168, no. 4, pp. 665–681, 2024.

[6] S. Anita *et al.*, "Classification cherry's coffee using k-nearest neighbor (knn) and artificial neural network (ann)," pp. 117–122, 2020.

[7] H. Singh, V. Sharma, and D. Singh, "Comparative analysis of proficiencies of various textures and geometric features in breast mass classification using k-nearest neighbor," *Visual Computing for Industry, Biomedicine, and Art*, vol. 5, no. 1, p. 3, 2022.

[8] F. Xiao, H. Wang, Y. Li, Y. Cao, X. Lv, and G. Xu, "Object detection and recognition techniques based on digital image processing and traditional machine learning for fruit and vegetable harvesting robots: An overview and review," *Agronomy*, vol. 13, no. 3, p. 639, 2023.

[9] A. Muneer and S. M. Fati, "Efficient and automated herbs classification approach based on shape and texture features using deep learning," *IEEE Access*, vol. 8, pp. 196747–196764, 2020.

[10] T. Turay and T. Vladimirova, "Toward performing image classification and object detection with convolutional neural networks in autonomous driving systems: A survey," *IEEE access*, vol. 10, pp. 14076–14119, 2022.

[11] M. M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," *Computation*, vol. 11, no. 3, p. 52, 2023.

[12] A.-A. Tulbure, A.-A. Tulbure, and E.-H. Dulf, "A review on modern defect detection models using dcnns–deep convolutional neural networks," *Journal of Advanced Research*, vol. 35, pp. 33–48, 2022.

[13] J. Liu and X. Wang, "Early recognition of tomato gray leaf spot disease based on mobilenetv2-yolov3 model," *Plant Methods*, vol. 16, no. 1, p. 83, 2020.

[14] R. Indraswari, R. Rokhana, and W. Herulambang, "Melanoma image classification based on mobilenetv2 network," *Procedia computer science*, vol. 197, pp. 198–207, 2022.

[15] Q. Zhu, H. Zhuang, M. Zhao, S. Xu, and R. Meng, "A study on expression recognition based on improved mobilenetv2 network," *Scientific Reports*, vol. 14, no. 1, p. 8121, 2024.

[16] P. Silva, E. Luz, G. Silva, G. Moreira, R. Silva, D. Lucio, and D. Menotti, "Covid-19 detection in ct images with deep learning: A voting-based scheme and cross-datasets analysis," *Informatics in medicine unlocked*, vol. 20, p. 100427, 2020.

[17] A. Rácz, D. Bajusz, and K. Héberger, "Effect of dataset size and train/test split ratios in qsar/qspr multiclass classification," *Molecules*, vol. 26, no. 4, p. 1111, 2021.

[18] V. Singh, M. Pencina, A. J. Einstein, J. X. Liang, D. S. Berman, and P. Slomka, "Impact of train/test sample regimen on performance estimate stability of machine learning in cardiovascular imaging," *Scientific reports*, vol. 11, no. 1, p. 14490, 2021.

[19] A. Mifthauddin, M. Lutfi, and Z. N. Saadah, "Comparison of mobilenetv2 and mobilenetv3 architectures in rice leaf disease classification using transfer learning," *Jurnal Mandiri IT*, vol. 14, no. 2, pp. 195–202, 2025.

[20] I. A. Usmani, M. T. Qadri, R. Zia, F. S. Alrayes, O. Saidani, and K. Dashtipour, "Interactive effect of learning rate and batch size to implement transfer learning for brain tumor classification," *Electronics*, vol. 12, no. 4, p. 964, 2023.

[21] N. Shome, R. Kashyap, and R. H. Laskar, "Detection of tuberculosis using customized mobilenet and transfer learning from chest x-ray image," *Image and Vision Computing*, vol. 147, p. 105063, 2024.

[22] N. Shome, R. Kashyap, and R. H. Laskar, "Detection of tuberculosis using customized mobilenet and transfer learning from chest x-ray image," *Image and Vision Computing*, vol. 147, p. 105063, 2024.

[23] W. Ma, X. Wang, S. X. Yang, X. Xue, M. Li, R. Wang, L. Yu, L. Song, and Q. Li, "Autonomous inspection robot for dead laying hens in caged layer house," *Computers and Electronics in Agriculture*, vol. 227, p. 109595, 2024.

[24] I. P. Sari, B. Warsito, and O. D. Nurhayati, "Automated classification of parasitic worm eggs based on transfer learning and fine-tuned cnn models.," *International Journal of Advanced Computer Science & Applications*, vol. 16, no. 5, 2025.

[25] A. Desiani, R. Primartha, H. Hanum, S. R. P. Dewi, B. Suprihatin, M. G. Al-Filambany, and M. Suedarmin, "Weighted voting ensemble learning of cnn architectures for diabetic retinopathy classification," *Jurnal Infotel*, vol. 16, no. 1, pp. 136–155, 2024.

[26] P. Irfansyah, Y. A. Purwanto, S. H. Wijaya, and N. Nahrowi, "Particle size detection of palm kernel cake from sieving based on images using convolutional neural network," *JURNAL INFOTEL*, vol. 17, no. 3, pp. 537–549, 2025.

[27] P.-C. Cheng and H.-H. K. Chiang, "Diagnosis of salivary gland tumors using transfer learning with fine-tuning and gradual unfreezing," *Diagnostics*, vol. 13, no. 21, p. 3333, 2023.

[28] V. Bhatnagar and P. P. Bansod, "Convolution neural network based multi-label disease detection using smartphone captured tongue images," *Applied Sciences*, vol. 14, no. 10, p. 4208, 2024.