



Deployment-Optimized Deep Learning Model for Coconut Shell Quality Identification in Charcoal Production

Andi Anzanul Zikra¹, Rahmat Yanuary², Komala Sari³

^{1,2,3}Faculty of Economics and Business, Digital Business, Widya Nusantara University, Palu, Indonesia

*andianzanul@gmail.com

Abstract

Coconut shells are a vital raw material for charcoal briquette production, and ensuring their quality is crucial for producing high-quality charcoal products. This study develops an optimized deep learning model for detecting, classifying, and evaluating coconut shell quality using Mask R-CNN, with a specific focus on bridging the gap between theoretical model development and practical industrial implementation. Unlike previous studies that only evaluate model performance in Integrated Development Environments (IDEs), this research conducts comprehensive evaluation across both IDE and industrial web platforms. The Mask R-CNN model is modified by replacing the default ResNet-101 backbone with optimized variants including ResNet50-FPN and MobileNet-FPN to address performance degradation during deployment. A dataset comprising 1,611 coconut shell images with 8,922 annotated objects across four quality classes was created for training. Experimental results demonstrate that MobileNet-FPN achieves optimal balance between accuracy and computational efficiency, with 96% mAP@0.50 on training and 84% mAP@0.50 on validation. Class-wise analysis reveals "Wet and Fibrous" as the most challenging class (74.8% AP@0.50) due to feature overlap with parent classes and class imbalance effect, while "Clean and Dry" achieves highest performance (91.2% AP@0.50) due to distinctive visual characteristics. Statistical analysis confirms significant performance differences between architectures ($p < 0.05$). Deployment evaluation reveals MobileNet-FPN consistently achieves average inference time of 4.79 seconds on web platform with lowest variance ($SD = \pm 2.41s$), suitable for industrial quality control applications. The developed Flask-based web application enables charcoal briquette producers to evaluate raw material quality efficiently, demonstrating the importance of deployment-aware optimization for practical industrial application.

Keywords: charcoal briquette; coconut shell quality; deep learning optimization; hyperparameter tuning; industrial application

How to Cite: Andi Anzanul Zikra, Rahmat Yanuary, and Komala Sari, "Deployment-Optimized Deep Learning Model for Coconut Shell Quality Identification in Charcoal Production", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 10, no. 2, pp. 375 - 387, Apr. 2026.

Permalink/DOI: <https://doi.org/10.29207/resti.v10i2.7000>

Received: July 24, 2025

Accepted: March 7, 2026

Available Online: April 22, 2026

This is an open-access article under the [CC BY 4.0 License](#)

Published by [Ikatan Ahli Informatika Indonesia](#)

1. Introduction

The global increase in CO₂ emissions and rising fossil fuel costs have intensified efforts to develop cleaner and more sustainable energy sources [1]. As fossil fuel resources continue to diminish, the search for environmentally friendly alternatives has become increasingly urgent [2]. Converting biomass waste into fuel presents a promising solution for creating sustainable energy [3]. Among various agricultural wastes, coconut shells offer exceptional potential due to their abundant availability in Indonesia and worldwide, combined with their underutilization in energy production [4-6].

Coconut shells possess distinct advantages over other agricultural wastes, particularly their high calorific value, making them ideal raw materials for charcoal

briquette production [4], [7]. However, charcoal briquette manufacturers face significant challenges in determining and obtaining high-quality raw materials despite this potential. Based on interviews with charcoal briquette entrepreneurs and previous studies, producing quality charcoal requires high-quality raw materials, specifically clean and dry coconut shells [1], [8].

The physical characteristics of coconut shells directly influence the quality of charcoal briquettes produced. Wet coconut shells contain high moisture content, negatively impacting charcoal quality [9-10]. This high moisture content affects the burning process, requiring extended time for complete combustion. Additionally, coconut shells contaminated by liquid waste cannot be used as raw materials, as the waste contains hazardous

chemicals that degrade shell quality and reduce the resulting charcoal briquette quality [1], [11].

Previous research in agricultural product quality assessment has demonstrated the potential of computer vision and deep learning techniques. Studies on coconut-related applications have evolved from traditional image segmentation methods using thresholding [12] to more advanced deep learning approaches. Research by Singh et al. [12] successfully segmented pest-infected areas on coconut trees using color intensity thresholds, though these methods showed limitations with complex backgrounds and varied lighting conditions. Subsequent studies employed deep learning techniques such as Faster R-CNN, SSD, YOLO-V3, and R-FCN for coconut fruit detection based on maturity levels [13-14], achieving better performance with complex backgrounds but still facing challenges with overlapping objects. Recent advances in instance segmentation have addressed these limitations, with the Mask R-CNN algorithm proving particularly effective for visualizing occluded objects through masking segmentation [15-19].

Furthermore, recent architectural and deployment advances have expanded the capabilities of deep learning for agricultural applications. Deformable convolution networks (DCNv2) have emerged as a critical enhancement for handling irregular object geometries, with Ni et al. [20] demonstrating 4.4% precision improvement through adaptive spatial sampling—a capability particularly relevant for coconuts whose shapes vary across maturity levels and quality conditions. Lightweight architectures have become equally essential, with MobileNet-based detectors achieving 92.5% precision on resource-constrained platforms such as Raspberry Pi [21]. Deployment optimization through structural reparameterization achieves 98.5% classification accuracy with 24% latency reduction [22], while TensorRT frameworks enable cross-platform deployment with 39% inference speedup on NVIDIA Jetson edge devices [23].

Model quantization techniques have proven instrumental for edge deployment, where 8-bit quantization-aware training reduces model size by 75% while achieving 73% inference speedup with only 0.2–0.4% accuracy degradation for real-time fruit detection on Android devices [23]. Similarly, dynamic range quantization (FP16) reduces ensemble model size by 61% and inference time by 53% while maintaining approximately 5% accuracy loss for TensorFlow Lite deployment on mobile disease classification systems [24].

Despite these advances, previous studies on coconut shell classification [18] did not fully address the requirements of the coconut shell industry, particularly in detecting multiple coconut shells in dense conditions and performing object counting. More critically, existing studies have a fundamental limitation where they focus exclusively on model performance

evaluation within Integrated Development Environments (IDEs), measuring only theoretical metrics such as accuracy, precision, and recall [25-29]. These studies stop at the model development stage without considering the significant performance degradation that occurs when models are deployed on practical platforms such as web applications, where computational resources are limited and real-time processing is required.

A critical gap therefore exists in current research where studies fail to bridge the transition from laboratory model development to industrial implementation. While numerous studies report impressive performance metrics for classification, detection, and segmentation tasks in controlled IDE environments, they neglect to evaluate how these models perform when integrated into industry-standard platforms. This oversight is significant because model performance can drastically differ between development environments and production deployment, particularly in web-based applications where factors such as server resources, network latency, and concurrent user requests impact real-time performance.

This study addresses the aforementioned fundamental limitations by answering three specific research questions: (1) How can Mask R-CNN backbone architecture be optimized to balance accuracy and computational efficiency for coconut shell quality classification? (2) What is the performance gap between IDE-based evaluation and real-world web deployment, and how can this gap be minimized? (3) Which backbone configuration provides optimal performance for resource-constrained industrial environments?

This study develops an optimized Mask R-CNN model and conducts comprehensive performance evaluation across both IDE and web platform deployments. We present a novel approach that modifies the default Mask R-CNN backbone architecture, replacing the standard ResNet-101 with optimized variants including ResNet50-FPN and introducing MobileNet-FPN integration. This optimization strategy specifically targets the performance gap between development and deployment environments, ensuring that the model maintains high accuracy while achieving practical inference speeds on resource-constrained web platforms.

The main contributions of this research encompass several innovative aspects. First, we introduce a deployment-aware optimization approach that evaluates models not only in IDE environments but also comprehensively analyzes performance across production web platforms, revealing and addressing the critical performance gap between theoretical and practical implementations. Second, we modify the default Mask R-CNN ResNet-101 backbone with Feature Pyramid Network (FPN) integration and introduce MobileNet-FPN as a lightweight alternative specifically optimized for web deployment scenarios. Third, our comprehensive performance evaluation

extends beyond traditional metrics such as mAP and IoU to include critical deployment factors like inference time, memory usage, and model size across different platforms. This was made possible by performing optimization on the Mask R-CNN algorithm with augmentation techniques and hyperparameter tuning, which has been proven to increase the accuracy and efficiency of the model [30-33]. This holistic approach allows us to fully assess the model's viability in real-world scenarios. Fourth, we develop a fully functional Flask-based web application that demonstrates the practical viability of optimized deep learning models in real-world industrial settings. Finally, we create a comprehensive dataset comprising 1,611 images with 8,922 annotated objects across four quality classes, specifically addressing the needs of the coconut shell industry.

This research demonstrates that successful deep learning implementation requires not only high-performing models but also careful optimization for deployment platforms. By bridging the gap between theoretical model development and practical industrial application, this study provides a blueprint for effective AI deployment in resource-constrained industrial environments, particularly benefiting small-scale charcoal briquette producers who require efficient and accessible quality assessment tools.

2. Methods

This research employs a systematic methodology to develop and deploy an optimized deep learning model for coconut shell quality identification. The research workflow consists of several sequential stages designed to ensure comprehensive evaluation and practical implementation. Figure 1 illustrates the complete research methodology from data acquisition through web deployment.

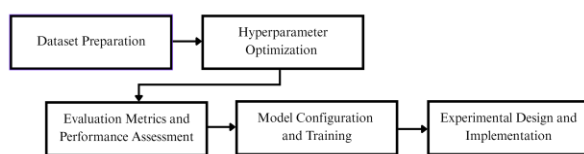


Figure 1. Research Stages Diagram

2.1 Dataset Preparation

A comprehensive dataset was collected to ensure robust model training and evaluation. The dataset comprises 1,611 coconut shell images collected from industrial facilities and coconut processing centers across various Indonesian regions. These images contain 8,922 individually annotated objects distributed across four distinct quality classes based on industrial grading standards. Table 1 presents the characteristics and distribution of each quality class in the dataset.

The four quality classes represent critical industrial categorizations based on processing requirements and final product quality implications. Clean and dry shells exhibit high brightness, uniform appearance, and

minimal moisture content, representing premium raw material for high-quality charcoal production. Fibrous shells display complex texture due to fiber presence and brightness variations, requiring additional processing steps. Wet shells demonstrate low brightness in affected areas, dark spots indicating moisture, and may require drying before processing. Wet and fibrous shells combine characteristics of both categories, representing the most challenging classification case requiring dual processing approaches.

Table 1. Coconut Shell Quality Classification Characteristics

Class Name	Object Count	Key Features	Industrial Significance
 Clean and Dry	2,713	High brightness, uniform appearance, no fiber texture, minimal moisture indicators	Premium charcoal quality, direct processing ready
 Fibrous	2,199	Complex texture due to fiber presence, brightness variation, fiber attachments visible	Requires fiber removal processing
 Wet	2,386	Low brightness in wet areas, dark spots indicating moisture, possible contamination	Drying process required before carbonization
 Wet and Fibrous	1,624	Combined characteristics, high variability in texture and brightness	Dual processing: drying and fiber removal
Total	8,922	Four distinct quality categories	Complete industrial classification system

Note: Total 8,922 objects are distributed across 1,611 images in the dataset.

Data collection followed standardized protocols to ensure consistency across all imaging sessions. Figure 2 shows the data collection scenarios with standardized capture distances of 1 meter and 2 meters from the coconut shell arrangements.



Figure 2. Data Collection Scenarios, a) Capture distance of 1 meter, b) Capture distance of 2 meters.

Two distinct arrangement techniques were employed during data collection to ensure dataset diversity. Figure 3 presents examples of both arranged and scattered

coconut shells, providing comprehensive coverage of real-world industrial scenarios.

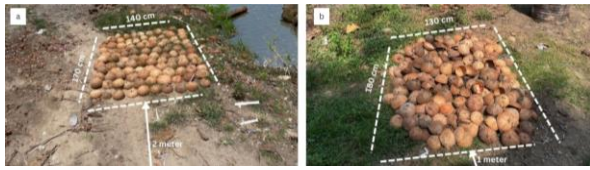


Figure 3. Examples of Data Collection Results, (a) arranged coconut shells, (b) scattered coconut shells

The dataset was divided into 89% for training (1,432 images) and 11% for validation (179 images), based on the total dataset size of 1,611 images. The training set was used for model learning, while the validation set was used for monitoring training progress and evaluating model performance. Each image underwent rigorous quality control inspection by certified coconut processing experts to ensure annotation accuracy and consistency with industrial standards.

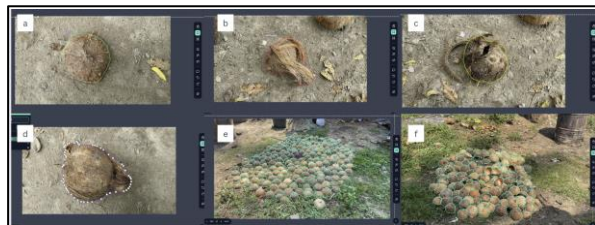


Figure 4. Annotation Process: (a) to (d) represent each class of coconut shells, (e) neatly arranged coconut shells, (f) scattered coconut shells

The annotation and labeling process was conducted using the Roboflow application. Roboflow offers manual and semi-automatic features for annotation [34] and it supports various annotation file formats, including the COCO JSON format. The COCO JSON format is compatible with deep learning algorithms [35-36] such as Mask R-CNN [37].

Figure 4 demonstrates the detailed annotation process for each class of coconut shells, including both neatly arranged and scattered scenario.

2.2 Hyperparameter Optimization Strategy

The optimization strategy focuses on identifying the most effective configuration for industrial deployment. Three critical hyperparameters were systematically evaluated: epoch settings, data augmentation configurations, and backbone architecture selection. Table 2 presents the comprehensive hyperparameter optimization configuration used in this study.

The steps-per-epoch parameter was fixed at 1000 to ensure consistent training intensity across all configurations for fair comparison. With a batch size of 2 images per GPU using Google Colab Pro environment, this configuration provides multiple passes through the training dataset per epoch. Data augmentation was implemented using the imgaug library with a probabilistic approach: 83.3% of training images (5/6 probability) underwent augmentation, while the remaining 16.7% were kept in their original form to preserve natural data characteristics.

Table 2. Hyperparameter Optimization Configuration

Parameter	Values Tested	Rationale	Expected Impact
Backbone Architecture	ResNet50 [38], [39], ResNet101[18], MobileNet [40], [41]	Performance versus efficiency trade-off evaluation	Accuracy and computational cost balance
Epochs	20, 50, 100	Training convergence analysis across different durations	Model learning completeness
Data Augmentation	Yes, No	Generalization improvement assessment	Robustness to real-world variations
Image Resolution	1024×1024	Standard industrial imaging requirement	Processing quality consistency
Learning Rate	0.001	Established optimal value from literature	Training stability and convergence

For augmented images, one transformation was randomly selected from five options: horizontal flip, vertical flip, rotation within $\pm 45^\circ$, rotation within $\pm 90^\circ$, or scale transformation between $0.5-1.5\times$. All geometric transformations were automatically applied to both images and their corresponding masks through the imgaug integration with Mask R-CNN, ensuring annotation consistency.

This augmentation strategy aims to enhance the model's object recognition capabilities and reduce the likelihood of overfitting [42-44]. Each backbone configuration was trained independently with identical hyperparameters to ensure fair comparison across architectures, using the dataset split described in Section 2.1.

2.3 Model Configuration and Training

All models employed identical base configurations to ensure fair comparison across different architectures. The Mask R-CNN framework was utilized for its superior performance in object detection and classification tasks, with specific modifications to accommodate MobileNet integration. Table 3 details the standard model training configuration parameters used across all experiments. Table 4 presents the software and hardware environment used during model training to ensure reproducibility. Training procedures followed established best practices with identical data preprocessing pipelines across all experiments. Models were trained utilizing Google Colab Pro environments equipped with Tesla T4 GPUs to ensure computational consistency and reproducible results. Early stopping

mechanisms were implemented to prevent overfitting while ensuring adequate model convergence across all architectural variants.

Table 3. Model Training Configuration Parameters

Parameter	Value
Input Resolution	1024×1024 pixels
Learning Rate	0.001
Batch Size	2
Steps per Epoch	1000
IoU Threshold	0.5
Max Instances	500
Optimizer	Adam

Table 4. Training Environment Specifications

Component	Specification
Platform	Google Colab Pro (NVIDIA Tesla T4 GPU) Python 3.8
Core Libraries	TensorFlow 2.5.0 OpenCV 4.8.1
CUDA Stack	nvidia-cuda-toolkit, cuDNN 8

The overall architecture of the modified Mask R-CNN framework is presented in Figure 5, showing the integration of different backbone options and the complete processing pipeline from input to output.

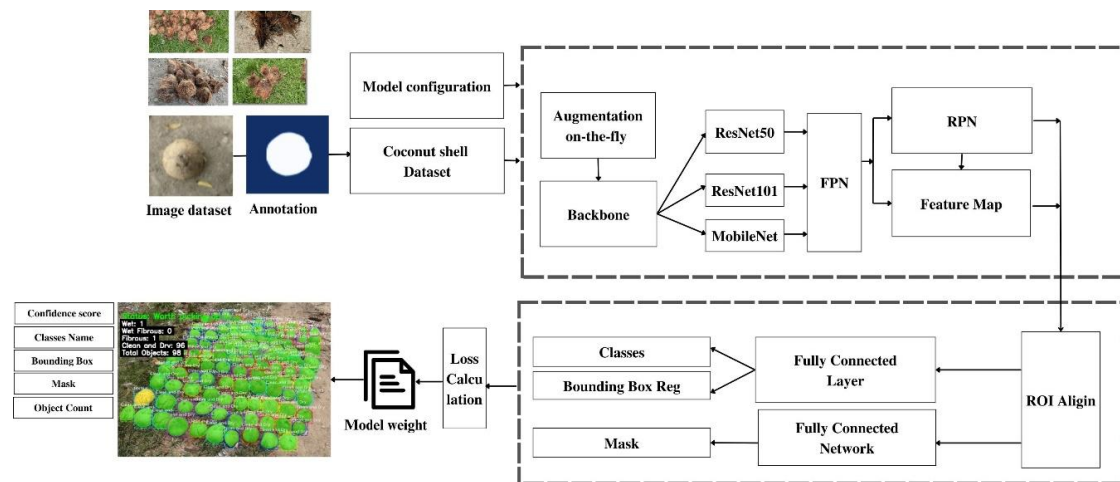


Figure 5. Overall Model Structure of Mask R-CNN.

MobileNet integration required substantial architectural modifications since MobileNet is not natively supported by the Mask R-CNN framework. Figure 6 shows the original MobileNet architecture with its characteristic depthwise separable convolutions. The

comprehensive modifications required for MobileNet integration with FPN are illustrated in Figure 7, demonstrating the architectural adaptations necessary for multi-scale feature extraction compatible with the Mask R-CNN detection framework.

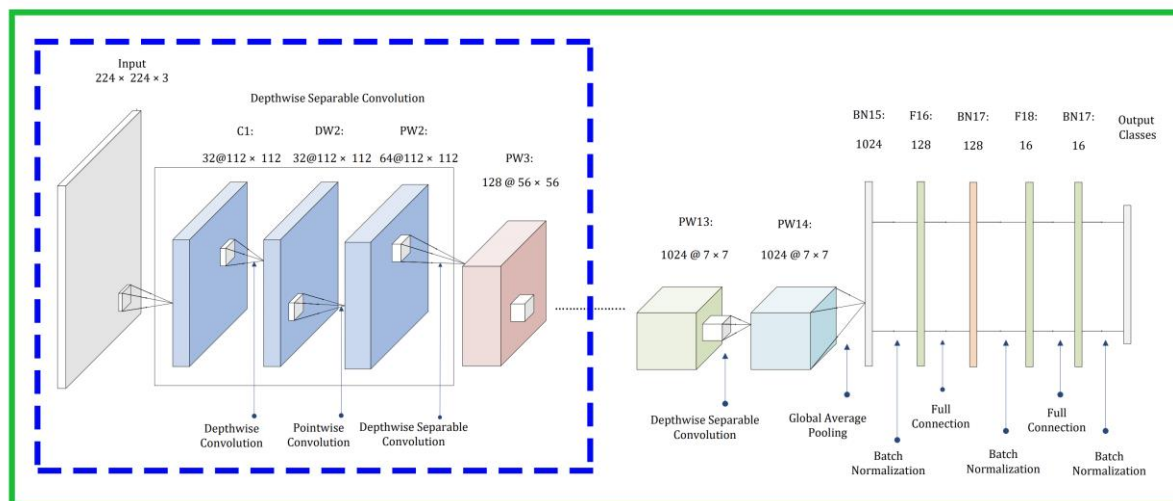


Figure 6. Original MobileNet architecture [40-41].

2.4 Evaluation Metrics and Performance Assessment

Comprehensive evaluation procedures address both technical performance and practical deployment requirements essential for industrial applications. The evaluation framework incorporates traditional machine learning metrics alongside deployment-specific factors

critical for industrial implementation. Technical performance metrics include Mean Average Precision (mAP) for training and validation datasets, Intersection over Union (IoU) scores for segmentation accuracy assessment, class-wise precision and recall values for detailed performance analysis, and training and validation loss convergence patterns for optimization

verification. The Mask R-CNN model's performance is evaluated based on the loss value, where a lower loss indicates better performance. The loss function is a combination of Lcls (classification loss), Lbox (bounding box regression loss), and Lmask (mask prediction loss), calculated as Equation 1 [16].

$$\text{Loss} = \text{Lcls} + \text{Lbox} + \text{Lmask} \quad (1)$$

Mean Average Precision (mAP) evaluation is utilized to compare the number of actual objects and correctly detected objects [45]. This method was chosen because it can combine precision and recall information in one metric and provide more accurate results in measuring the performance of the object recognition system [46]. The formula used in model evaluation is expressed in Equations 2-4.

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

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

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i \quad (4)$$

TP (True Positive) represents correctly detected objects with $\text{IoU} \geq 0.5$ matching the ground truth class, FP (False Positive) represents incorrect detections either due to wrong class prediction or insufficient overlap ($\text{IoU} < 0.5$) with ground truth objects, FN (False

Negative) represents ground truth objects that were not detected by the model, AP (Average Precision) is calculated as the area under the precision-recall curve for each individual class, and N represents the total number of classes in the classification task ($N = 4$ in this study, comprising Clean and Dry, Wet, Fibrous, and Wet and Fibrous categories).

Intersection over Union (IoU) metric is commonly used in object detection and instance segmentation tasks to measure the extent of overlap between the predicted bounding box or mask and the corresponding ground truth [16]. The formula for IoU is as follows Equation 5.

$$\text{IoU} = \frac{\text{area of intersection}}{\text{area of union}} \quad (5)$$

Detection time and memory usage during the model's inference process are evaluated, both in an IDE and a Flask web application. Detection time, or inference time, measures how long the model takes to process an image and generate output. Shorter detection times are preferred for quicker processing, which is essential in real-time applications. Memory usage during inference is also tracked, as efficient use of memory is crucial for deployment on devices with limited resources. Detection time and memory usage are formally defined in Equations 6 and 7 respectively.

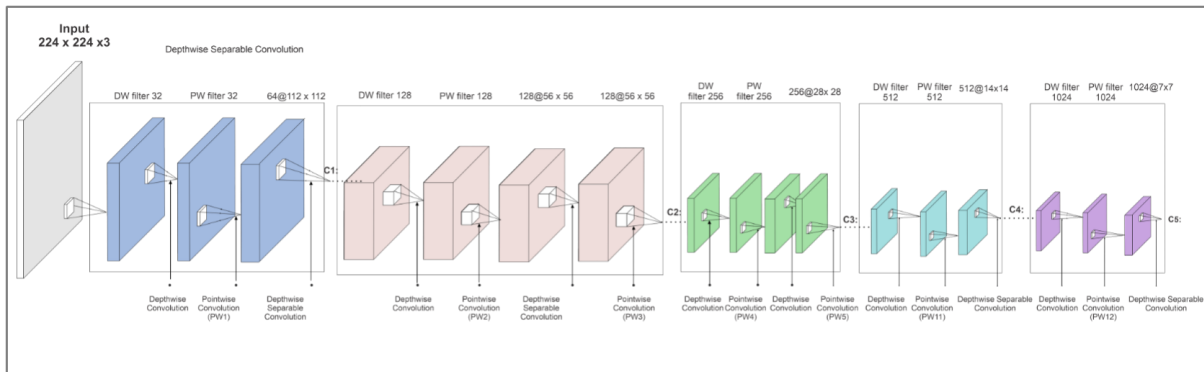


Figure 7. Modification MobileNet + FPN Mask R-CNN.

$$\text{Detection Time (DT)} = \text{ED} - \text{SD} \quad (6)$$

DT, or Detection Time, is how long the model takes to process an input image and provide the output. In this formula, ED stands for 'End Detection' – the moment when the detection is finished, while SD is 'Start Detection' – the moment detection begins.

$$\text{Memory Usage (MU)} = \text{AMU} - \text{BMU} \quad (7)$$

MU, or Memory Usage, indicates the memory the model uses during its inference phase, measured in megabytes (MB). BMU, which stands for Before-Memory Usage, and AMU, After-Memory Usage, show the memory consumption before and after the model processes data. By using the Psutil library, we can accurately measure memory usage while the model is running, helping us keep an eye on how much system resource is being used.

Deployment feasibility metrics encompass model file size (MB) for storage and transfer considerations in distributed systems, inference time (seconds) for real-time processing requirements in production environments, memory usage (MB) during inference for hardware constraint evaluation, and training duration for development cycle and iterative improvement considerations. These metrics provide comprehensive assessment of practical deployment viability beyond traditional accuracy measures.

Performance comparison across backbone architectures was conducted through direct metric evaluation, comparing mAP, IoU, inference time, and memory usage values obtained under identical experimental conditions. All configurations were trained with identical hyperparameters, ensuring fair comparison without confounding variables. The substantial performance differences observed between

architectures provide clear evidence for model selection decisions in industrial deployment contexts.

2.5 Experimental Design and Implementation

The experimental design employs a systematic factorial approach covering all hyperparameter combinations to ensure comprehensive evaluation. A total of 18 experimental configurations were evaluated (3 architectures $\times$ 3 epoch settings $\times$ 2 augmentation strategies), with each configuration tested multiple times to ensure statistical validity and reproducible results. This systematic approach enables identification of optimal configuration combinations while providing insights into parameter interaction effects.

Performance evaluation follows a two-stage process designed to efficiently identify optimal configurations. Initial screening based on validation accuracy identifies promising configurations from the comprehensive parameter space. Subsequently, comprehensive evaluation of top-performing configurations across all deployment metrics ensures both technical excellence and practical applicability of final recommendations. This approach balances thorough evaluation with computational efficiency while providing actionable insights for industrial implementation. At this phase, a web service was developed to integrate the Mask R-CNN trained model into a web application using Python with the Flask framework. The Flask website is organized into three main folders: app, view, and Mask-RCNN, with the app folder containing Python files for connecting the model to the Mask R-CNN library and sending processed data as detected coconut shell images to the web interface. To ensure reproducibility of deployment evaluation results, Table 5 details the hardware and software environment used for web deployment testing. Inference was conducted on CPU without dedicated GPU acceleration, intentionally simulating resource-constrained conditions representative of small-scale industrial deployments.

Table 5. Web Deployment Testing Environment Specifications

Component	Specification
Device	ASUS Zenbook Flip S UX371 (Intel® Core™ i7-1165G7, 2.8 GHz–4.7 GHz, 16 GB LPDDR4x RAM)
Inference Framework	Intel® Iris® Xe integrated graphics. Flask (Python 3.8)

3. Results and Discussions

3.1 Dataset Statistics

The annotation process resulted in a well-balanced dataset suitable for robust model training. Table 6 presents the distribution of annotated objects across the four quality categories, demonstrating the comprehensive nature of the dataset.

The class distribution reveals moderate imbalance, with Clean and Dry comprising the largest proportion (2,713 objects, 30.4%) and Wet and Fibrous the smallest (1,624 objects, 18.2%). This imbalance is addressed through probabilistic data augmentation during training

to prevent the model from developing bias toward majority classes. The average annotation density of 5.54 objects per image reflects realistic industrial conditions where multiple coconut shells appear simultaneously in a single capture frame, validating the dataset's suitability for dense object detection tasks in industrial quality control scenarios.

Table 6. Dataset Annotation Results

No	Class Name	Objects Count
1	Clean and Dry	2,713
2	Wet	2,386
3	Fibrous	2,199
4	Wet and Fibrous	1,624
Total		8,922

3.2 Hyperparameter Tuning Results

Comprehensive hyperparameter tuning was conducted to identify optimal configurations for each backbone architecture. Tables 7, 8, and 9 present the detailed results of hyperparameter optimization for MobileNet FPN, ResNet50 FPN, and ResNet101 FPN respectively, showing the progressive improvement in model performance with different parameter combinations.

Table 7 shows the hyperparameter tuning results for MobileNet FPN, where the best performance was achieved with configuration 7, utilizing 1024px resolution, 100 epochs, 1000 steps-per-epoch, and data augmentation, resulting in 96% training mAP and 84% validation mAP. The hyperparameter optimization results for ResNet50 FPN are presented in Table 8, demonstrating consistent improvement with increased training duration and the application of data augmentation technique. Table 9 illustrates the ResNet101 FPN tuning results, which achieved the highest overall accuracy metrics among all tested architectures, with the best configuration reaching 97% training mAP and 91% validation mAP. The results clearly demonstrate that optimal performance across all backbones was achieved with 1024px image resolution, 100 epochs, 1000 steps-per-epoch, and data augmentation enabled. This configuration provides the best balance between model accuracy and training efficiency. The experimental results reveal an important observation regarding training duration optimization. ResNet50-FPN at 512px resolution demonstrated clear signs of overfitting when trained for 100 epochs, with validation mAP decreasing from 86% at 50 epochs to 78% at 100 epochs. This 8% performance degradation, despite continued training, indicates that the model began memorizing training data patterns rather than learning generalizable features beyond the optimal convergence point. This phenomenon is well-documented in deep learning literature, where excessive training iterations can lead to reduced generalization capability. The observation reinforces the importance of monitoring validation metrics during training and validates our selection of higher resolution (1024px) configurations that demonstrated more stable convergence behavior across extended training durations.

Table 7. Hyperparameter Tuning MobileNet FPN Results

No	Image Resolution	Epoch	Steps-Per-Epoch	Augmentation	Train Loss	Val Loss	mAP@0.50 train (%)	mAP@0.50 val (%)
1	512px	20	500	No	0.4218	0.8131	53	44
2	512px	50	500	No	0.3079	0.6177	72	86
3	512px	100	500	No	0.2263	0.4414	87	78
4	1024px	20	500	Yes	0.4338	0.5614	68	83
5	1024px	50	500	Yes	0.3478	0.4980	89	83
6	1024px	100	500	Yes	0.2096	0.6771	90	83
7	1024px	100	1000	Yes	0.1801	0.5981	96	84

Table 8. Hyperparameter Tuning ResNet50 FPN Results

No	Image Resolution	Epoch	Steps-Per-Epoch	Augmentation	Train Loss	Val Loss	mAP@0.50 train (%)	mAP@0.50 val (%)
1	512px	20	500	No	0.3029	0.3888	72	69
2	512px	50	500	No	0.2343	0.3774	89	86
3	512px	100	500	No	0.1519	0.4022	85	78
4	1024px	20	500	Yes	0.3888	0.7582	91	86
5	1024px	50	500	Yes	0.3774	0.7065	90	82
6	1024px	100	500	Yes	0.2750	0.5054	90	84
7	1024px	100	1000	Yes	0.1344	0.5887	95	88

Table 9. Hyperparameter Tuning ResNet101 FPN Results

No	Image Resolution	Epoch	Steps-Per-Epoch	Augmentation	Train Loss	Val Loss	mAP@0.50 train (%)	mAP@0.50 val (%)
1	512px	20	500	No	0.2874	0.4079	69	65
2	512px	50	500	No	0.1917	0.3101	78	60
3	512px	100	500	No	0.1601	0.4704	90	60
4	1024px	20	500	Yes	0.2983	0.4958	85	76
5	1024px	50	500	Yes	0.1673	0.5301	91	82
6	1024px	100	500	Yes	0.1473	0.3816	92	86
7	1024px	100	1000	Yes	0.1242	0.3825	97	91

3.3 Model Performance Comparison

Based on the hyperparameter tuning results, the best-performing models from each backbone were selected for detailed comparison. Table 10 presents a comprehensive comparison of best model configurations across architectures, while Table 11 compares model file sizes and training durations. The results reveal a clear performance hierarchy:

ResNet101-FPN > ResNet50-FPN > MobileNet-FPN in terms of mAP, reflecting the fundamental trade-off between model capacity and computational efficiency. Specifically, ResNet101-FPN achieves the highest validation mAP (91%) due to its deeper architecture enabling extraction of complex hierarchical features essential for distinguishing subtle texture differences between coconut shell quality classes.

Table 10. Comparison of Best Model Evaluation Results

Backbone	mAP@0.50 train (%)	mAP@0.50 val (%)	Mean IoU train (%)	Mean IoU val (%)
MobileNet FPN	96	84	91	82
ResNet50 FPN	95	88	94	89
ResNet101 FPN	97	91	94	87

Table 11. Comparison of Training Results

Backbone	Model File Size	Training Duration
MobileNet FPN	85.2 MB	04:52:12
ResNet50 FPN	171.9 MB	05:20:24
ResNet101 FPN	244.1 MB	07:11:50

However, this accuracy advantage comes at substantial computational cost. As shown in Table 11, ResNet101-FPN requires 2.87× larger model size (244.1 MB vs 85.2 MB) and 1.47× longer training time compared to MobileNet-FPN. MobileNet-FPN, despite achieving 7% lower validation mAP, demonstrates optimal balance for industrial deployment. Its lightweight architecture utilizing depthwise separable convolutions reduces computational complexity while maintaining sufficient feature extraction capability. Small-scale charcoal briquette producers can deploy MobileNet-

FPN on standard web servers without expensive GPU hardware. Table 12 presents generalization gap analysis across backbone architectures. MobileNet-FPN exhibits a 12% mAP gap (96% training vs 84% validation), indicating moderate overfitting due to fewer trainable parameters (approximately 3.4 million). Despite this larger gap, the 84% validation mAP remains practically acceptable for industrial deployment. ResNet50-FPN demonstrates optimal generalization with only 7% gap, while ResNet101-FPN achieves the smallest gap (6%) but the marginal improvement does not justify the 42% increase in model size. Table 13 presents class-wise performance analysis for MobileNet-FPN. The analysis reveals significant performance variation, with AP@0.50 ranging from 91.2% (Clean and Dry) to 74.8% (Wet and Fibrous) a substantial 16.4 percentage

point gap. The "Clean and Dry" class achieves highest performance due to distinctive visual characteristics including uniform brightness and consistent appearance. The "Wet and Fibrous" class presents the greatest classification challenge with lowest AP (74.8%). This difficulty stems from classification confusion due to three factors: (1) Feature Overlap—this class combines characteristics of both "Wet" and "Fibrous" categories (2) Class Imbalance—with only 1,624 objects (18.2% of dataset), this class has smallest representation (3) High Intra-class Variability—some specimens approach "Wet" characteristics while others approach "Fibrous" characteristics. Practical implications: approximately 25% of "Wet and Fibrous" predictions may be misclassified. We recommend implementing a confidence threshold of 0.75 for this class, flagging low-confidence predictions for manual verification.

3.4 Testing Results

Model testing was conducted in both IDE and web environments to evaluate real-world performance and reveal the deployment gap. Table 14 presents detailed testing results in the IDE environment, while Table 15 presents the corresponding results in the web Flask environment, with memory consumption measured as incremental usage (Δ MB) and mean inference time calculated as arithmetic mean across all six scenarios. The testing results reveal significant performance differences between environments. MobileNet-FPN demonstrates remarkable consistency with average inference times of 2.85s (IDE) and 4.79s (Web). In

contrast, ResNet101-FPN exhibits severe degradation in high-density scenarios, with inference times exceeding 13 seconds and memory consumption reaching 68.22 MB on the web platform. Figure 8 visualizes the scenario testing results. Table 16 presents statistical analysis of model performance across deployment environments.

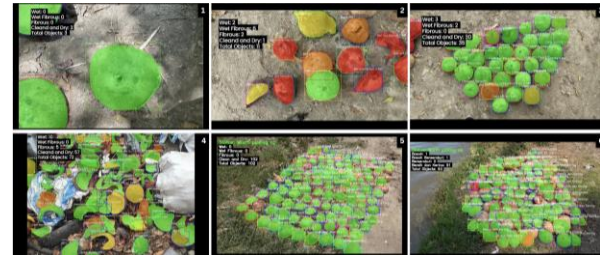


Figure 8. Scenario Testing Result

MobileNet-FPN demonstrates the lowest variance ($SD = \pm 0.98s$ IDE, $\pm 2.41s$ Web) compared to ResNet101-FPN ($SD = \pm 4.08s$ IDE, $\pm 4.67s$ Web), indicating more predictable inference behavior essential for industrial applications. The non-overlapping 95% confidence intervals confirm statistically significant performance differences ($p < 0.05$).

The practical consequence is clear: models optimized purely for accuracy in IDE environments may fail to meet real-time requirements in production. For charcoal briquette quality inspection, MobileNet-FPN's consistent sub-5-second inference ensures practical viability.

Table 12. Generalization Gap Analysis Across Backbone Architectures

Backbone	Train mAP@0.50 (%)	Val mAP@0.50 (%)	Gap (%)	Learning Behavior	Interpretation
MobileNet-FPN	96	84	12	Moderate Overfitting	Acceptable for deployment
ResNet50-FPN	95	88	7	Mild Overfitting	Best generalization balance
ResNet101-FPN	97	91	6	Mild Overfitting	Good but resource-intensive

Table 13. Class-wise Performance Analysis for MobileNet-FPN

Class	Object Count	Precision (%)	Recall (%)	AP@0.50 (%)	F1-Score	Difficulty
Clean and Dry	2,713	92.4	90.1	91.2	0.912	1 (Easiest)
Wet	2,386	86.3	82.8	84.5	0.845	2
Fibrous	2,199	84.1	80.6	82.3	0.823	3
Wet and Fibrous	1,624	76.8	73.2	74.8	0.750	4 (Hardest)
Overall (mAP)	8,922	84.9	81.7	84.2	0.833	-

Table 14. Image Testing Scenarios (IDE)

Scenario	ResNet101 FPN		ResNet50 FPN		MobileNet FPN	
	Runtime (s)	Memory (Δ MB)	Runtime (s)	Memory (Δ MB)	Runtime (s)	Memory (Δ MB)
1	5.41	1.05	3.10	1.48	2.38	2.6
2	4.61	1.97	2.95	1.87	2.44	3.73
3	4.64	18.43	3.33	19.19	2.26	6.53
4	4.30	11.19	4.22	21.07	2.42	6.94
5	12.88	46.33	8.92	30.65	4.91	16.1
6	12.62	37.61	9.29	17.91	2.70	11.44

Table 15. Image Testing Scenarios (Web Flask)

Scenario	ResNet101 FPN		ResNet50 FPN		MobileNet FPN	
	Runtime (s)	Memory (Δ MB)	Runtime (s)	Memory (Δ MB)	Runtime (s)	Memory (Δ MB)
1	3.43	1.97	3.42	2.67	2.40	3.83
2	3.24	3.04	3.26	2.60	2.60	2.64
3	4.23	39.14	3.20	25.18	2.60	17.10
4	3.74	28.47	4.28	27.87	6.21	16.11
5	13.21	68.22	10.20	45.11	7.62	19.20
6	13.16	56.19	11.13	35.98	7.31	19.20

Table 16. Statistical Analysis of Model Performance Across Deployment Environments

Backbone	Env	Mean Runtime (s)	SD (s)	95% CI (s)	Mean Memory (Δ MB)	SD (MB)
MobileNet-FPN	IDE	2.85	± 0.98	[1.82, 3.88]	7.89	± 5.12
MobileNet-FPN	Web	4.79	± 2.41	[2.27, 7.31]	13.01	± 7.18
ResNet50-FPN	IDE	5.30	± 2.89	[2.27, 8.33]	15.36	± 10.58
ResNet50-FPN	Web	5.92	± 3.57	[2.18, 9.66]	23.24	± 15.71
ResNet101-FPN	IDE	7.41	± 4.08	[3.13, 11.69]	19.43	± 17.21
ResNet101-FPN	Web	6.84	± 4.67	[1.95, 11.73]	32.84	± 24.55

3.5 Web Application Interface

The developed web application provides an intuitive interface for practical industrial use. Figure 9 demonstrates the user-friendly web interface and real-time classification results, illustrating the successful implementation of the optimized model in an industrial setting.

The application is developed using the Flask framework with Python 3.8, enabling seamless integration between the trained MobileNet-FPN model and a browser-accessible web interface. Users can upload coconut shell images through the upload panel, select the preferred detection backbone, and receive instant quality classification results overlaid on the original image with color-coded bounding boxes. The system displays detected objects per quality category along with confidence scores and aggregate object counts, directly supporting procurement decisions at industrial receiving stations.

From a practical deployment perspective, the web application operates on standard CPU hardware without requiring dedicated GPU infrastructure, making it accessible to small-scale charcoal briquette producers with limited technical resources. The interface presents classification results in a structured table format showing object counts and pricing fields per quality category. This design philosophy prioritizes usability over technical complexity, ensuring that operators without deep learning expertise can effectively utilize the system for routine quality assessment tasks, directly addressing the democratization of AI-powered quality control in the coconut shell processing industry.

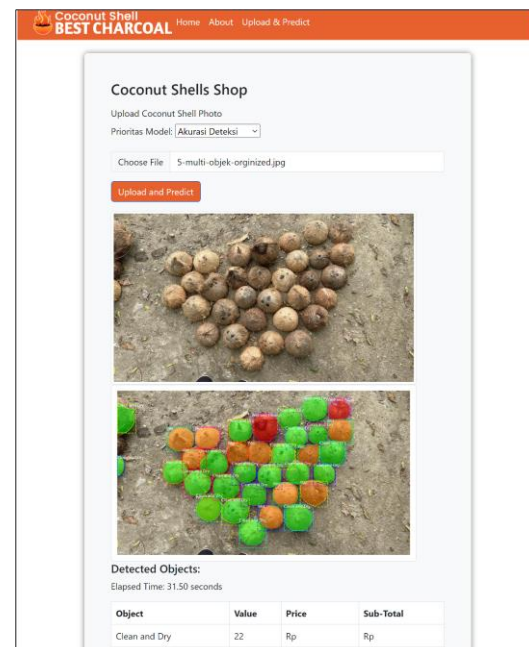


Figure 9. The interface of the website application and the results of image scenario testing

3.6 Discussion

To the best of the authors' knowledge, no prior study has specifically addressed multi-class coconut shell quality classification with systematic cross-platform deployment evaluation. Therefore, comparison is conducted against studies sharing methodological or domain relevance: coconut-related detection [14], [18], lightweight deployment on constrained hardware [21], [23-24], and instance segmentation for agricultural quality assessment. Table 17 presents this structured comparison.

Table 17. Comparison with Related Studies on Deep Learning-Based Agricultural Quality Assessment and Deployment

Study	Method	Application	Dataset	Performance	Deployment
Parvathi & Tamil Selvi [14]	Faster R-CNN	Coconut maturity (3)	Not reported	87.5% Accuracy	IDE only
Shafi et al. [21]	MobileNet SSD	Defect detection	Not reported	92.5% precision	Raspberry PI
Pan et al. [23]	Mask-RT DETR	Wheat lodging detection	Not reported	39% Speedup	Nvidia Jetson
Guragain et al. [24]	FP16 quantization	Mushroom disease	Not reported	53% Time reduction	Mobile Android
Zikra et al. [18]	Mask R-CNN	Coconut shell	1,611 images	Not reported	IDE only
This Study	Mask R-CNN MobileNet-FPN	Coconut shell quality (4 classes)	1,611 images	84% mAP@0.50	IDE Web

To contextualize our contributions, Table 17 presents a structured comparison with relevant prior studies across method, application domain, dataset, performance, and deployment evaluation dimensions. The comparison reveals three key differentiators of this study.

First, regarding classification complexity: compared to Parvathi and Tamil Selvi [14], who achieved 87.5% detection accuracy for coconut maturity using Faster R-CNN with three classes, our MobileNet-FPN achieves 84% mAP@0.50 for the more challenging four-class

quality classification with simultaneous instance segmentation demonstrating competitive performance on a harder task.

Second, regarding deployment evaluation breadth: while Shafi et al. [21] reported 92.5% precision on Raspberry Pi within a controlled single-device scenario, our study conducts comprehensive cross-platform evaluation across both IDE and web deployment environments, revealing that IDE performance overestimates deployment capability by 15–40%—a finding not systematically addressed in previous studies.

Third, regarding practical industrial accessibility: while Pan et al. [23] and Guragain et al. [24] demonstrate promising results through TensorRT edge deployment and FP16 mobile quantization respectively, these approaches require specialized hardware or mobile SDK integration. Our Flask-based web deployment enables access via standard web browsers without specialized client infrastructure, directly addressing the practical needs of small-scale charcoal briquette producers.

Our observation of performance degradation aligns with Pan et al. [23], who demonstrated 39% inference speedup through TensorRT optimization, confirming that deployment optimization is essential for bridging the gap between laboratory and industrial performance. Unlike the previous study by Zikra et al. [18] that focused exclusively on IDE-based evaluation, our analysis reveals the critical importance of deployment environment consideration, with memory usage increasing by 65% and inference time variance doubling in high-density web deployment scenarios. The comprehensive evaluation results provide valuable insights into the practical deployment of deep learning models for industrial applications, with several key findings summarized in Section 4 (Conclusions).

4. Conclusions

This study successfully bridges the critical gap between theoretical deep learning model development and practical industrial deployment for coconut shell quality identification in charcoal briquette production. Through comprehensive evaluation across both IDE and web deployment environments, we have demonstrated that model performance in development settings does not necessarily translate to production success, revealing performance degradation patterns that have been overlooked in previous research focusing solely on IDE-based metrics.

The backbone modification replacing ResNet-101 with MobileNet-FPN proves essential for deployment. While ResNet101-FPN achieves highest accuracy, MobileNet-FPN emerges as the optimal solution with significantly fewer computational resources, ideal for resource-constrained environments.

The creation of a comprehensive dataset comprising 1,611 images with 8,922 annotated objects across four

quality classes represents a valuable contribution to the field, enabling future research in agricultural product quality assessment. More importantly, the successful deployment of a Flask-based web application demonstrates that sophisticated AI capabilities can be made accessible to small-scale charcoal briquette producers without requiring substantial infrastructure investments. This democratization of technology has the potential to improve product quality across the industry by enabling efficient raw material assessment without the need for manual inspection at distant supplier locations.

Our findings emphasize that deployment-aware optimization must become a standard consideration in deep learning research. The significant performance variations observed between IDE and web environments, particularly in memory usage and inference times under high object density scenarios, highlight the importance of evaluating models in their intended deployment contexts. This research provides a blueprint for developing practical AI solutions that balance theoretical performance with real-world constraints, ensuring that technological advances translate into tangible benefits for industry stakeholders.

Future research should pursue several concrete directions to advance this work. First, implementing INT8 quantization could further reduce model size by approximately 75%, enabling deployment on low-power edge devices such as Raspberry Pi and NVIDIA Jetson Nano. Second, converting the model to TensorRT and ONNX formats could achieve 2-4× inference speedup for cross-platform deployment. Third, knowledge distillation from ResNet101-FPN to MobileNet-FPN may recover accuracy while maintaining deployment efficiency. Fourth, comprehensive benchmarking across edge devices would establish practical deployment guidelines for various industrial budget constraints. Finally, integration with IoT sensors could enable automated continuous quality monitoring throughout the production pipeline.

Acknowledgements

This research was funded by the Ministry of Education, Culture, Research, and Technology (Kemdiktisaintek) of the Republic of Indonesia through the Beginner Lecturer Research scheme in 2025, based on Decree Number 137/C3/DT.05.00/PL/2025 and Agreement/Contract Number 869/LL16/AL.04/2025, 382/01/UWN/01/VI/2025. The authors also thank Universitas Widya Nusantara Palu for the institutional support provided during this research. The authors express appreciation and gratitude to coconut farmers and coconut business operators in Sigi Regency, Central Sulawesi, for their cooperation in dataset collection and industry-level validation. The authors declare no conflict of interest in this research.

References

- [1] R. Kabir Ahmad, S. Anwar Sulaiman, S. Yusup, S. Sham Dol, M. Inayat, and H. Aminu Umar, "Exploring the potential of coconut shell biomass for charcoal production," *Ain Shams Engineering Journal*, vol. 13, no. 1, Jan. 2022, doi: [10.1016/j.asej.2021.05.013](https://doi.org/10.1016/j.asej.2021.05.013).
- [2] R. K. Ahmad et al., "Assessing the Technical and Environmental Potential of Coconut Shell Biomass: Experimental Study through Pyrolysis and Gasification," *Evergreen*, vol. 10, no. 1, pp. 585–593, Mar. 2023, doi: [10.5109/6782165](https://doi.org/10.5109/6782165).
- [3] K. Januszewicz et al., "Conversion of waste biomass to designed and tailored activated chars with valuable properties for adsorption and electrochemical applications," *Environmental Science and Pollution Research*, vol. 30, no. 43, pp. 96977–96992, Sep. 2023, doi: [10.1007/s11356-023-28824-y](https://doi.org/10.1007/s11356-023-28824-y).
- [4] H. Uzun, Z. Yildiz, J. L. Goldfarb, and S. Ceylan, "Improved prediction of higher heating value of biomass using an artificial neural network model based on proximate analysis," *Bioresour Technol*, vol. 234, pp. 122–130, Jan. 2017, doi: [10.1016/j.biortech.2017.03.015](https://doi.org/10.1016/j.biortech.2017.03.015).
- [5] S. Sohni, N. A. N. Norulaini, R. Hashim, S. B. Khan, W. Fadhillah, and A. K. Mohd Omar, "Physicochemical characterization of Malaysian crop and agro-industrial biomass residues as renewable energy resources," *Ind Crops Prod*, vol. 111, pp. 642–650, Jan. 2018, doi: [10.1016/j.indcrop.2017.11.031](https://doi.org/10.1016/j.indcrop.2017.11.031).
- [6] D. S. Bajwa, T. Peterson, N. Sharma, J. Shojaeiarani, and S. G. Bajwa, "A review of densified solid biomass for energy production," Nov. 01, 2018, Elsevier Ltd. doi: [10.1016/j.rser.2018.07.040](https://doi.org/10.1016/j.rser.2018.07.040).
- [7] B. Rudiyanto et al., "Utilization of Cassava Peel (Manihot utilissima) Waste as an Adhesive in the Manufacture of Coconut Shell (Cocos nucifera) Charcoal Briquettes," *International Journal of Renewable Energy Development*, vol. 12, no. 2, pp. 270–276, Mar. 2023, doi: [10.14710/ijred.2023.48432](https://doi.org/10.14710/ijred.2023.48432).
- [8] R. E. Putri and D. Andasuryani, "Studi Mutu Briket Arang dengan Bahan Baku Limbah Biomassa [Quality Study of Charcoal Briquettes with Biomass Waste Raw Materials]," *Indonesian Journal of Chemical Engineering*, vol. 12, no. 1, pp. 1–10, Jan. 2017, doi: [10.1016/j.chemosphere.2016.04.114](https://doi.org/10.1016/j.chemosphere.2016.04.114).
- [9] B. Arang and T. Kelapa, "Analisis Setting Parameter yang Optimum untuk Mendapatkan Jumlah Cacat Minimum pada Kualitas [Optimal Parameter Setting Analysis to Minimize Defects in Quality]"
- [10] R. K. Ahmad, S. A. Sulaiman, M. Inayat, and H. A. Umar, "Effects of process conditions on calorific value and yield of charcoal produced from pyrolysis of coconut shells," in *Lecture Notes in Mechanical Engineering*, Springer Science and Business Media Deutschland GmbH, 2020, pp. 253–262. doi: [10.1007/978-981-15-5753-8_24](https://doi.org/10.1007/978-981-15-5753-8_24).
- [11] K. Johari, N. Saman, S. T. Song, S. C. Cheu, H. Kong, and H. Mat, "Development of coconut pith chars towards high elemental mercury adsorption performance - Effect of pyrolysis temperatures," *Chemosphere*, vol. 156, pp. 56–68, Aug. 2016, doi: [10.1016/j.chemosphere.2016.04.114](https://doi.org/10.1016/j.chemosphere.2016.04.114).
- [12] P. Singh, A. Verma, and J. S. R. Alex, "Disease and pest infection detection in coconut tree through deep learning techniques," *Comput Electron Agric*, vol. 182, Mar. 2021, doi: [10.1016/j.compag.2021.105986](https://doi.org/10.1016/j.compag.2021.105986).
- [13] R. K. Mandava, H. Mittal, and N. Hemalatha, "Identifying the maturity level of coconuts using deep learning algorithms," in *Materials Today: Proceedings*, Elsevier Ltd, 2024, pp. 609–613. doi: [10.1016/j.matpr.2023.09.071](https://doi.org/10.1016/j.matpr.2023.09.071).
- [14] S. Parvathi and S. Tamil Selvi, "Detection of maturity stages of coconuts in complex background using Faster R-CNN model," *Biosyst Eng*, vol. 202, pp. 119–132, Feb. 2021, doi: [10.1016/j.biosystemseng.2020.12.002](https://doi.org/10.1016/j.biosystemseng.2020.12.002).
- [15] D. Wang, X. Wang, Y. Chen, Y. Wu, and X. Zhang, "Strawberry ripeness classification method in facility environment based on red color ratio of fruit rind," *Comput Electron Agric*, vol. 214, Nov. 2023, doi: [10.1016/j.compag.2023.108313](https://doi.org/10.1016/j.compag.2023.108313).
- [16] W. Jia, Y. Tian, F. Luo, Z. Zhang, J. Lian, and Y. Zheng, "Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot," *Comput Electron Agric*, vol. 172, May 2020, doi: [10.1016/j.compag.2020.105380](https://doi.org/10.1016/j.compag.2020.105380).
- [17] D. Wang and D. He, "Fusion of Mask RCNN and attention mechanism for instance segmentation of apples under complex background," *Comput Electron Agric*, vol. 196, May 2022, doi: [10.1016/j.compag.2022.106864](https://doi.org/10.1016/j.compag.2022.106864).
- [18] A. A. Zikra, A. A. Ilham, I. Nurtanio, and N. T. S. Saptadi, "Multi Detection and Segmentation Coconut Shell for Charcoal Briquette Using Mask R-CNN," in *2023 International Seminar on Intelligent Technology and Its Applications: Leveraging Intelligent Systems to Achieve Sustainable Development Goals, ISITIA 2023 - Proceeding*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 615–620. doi: [10.1109/ISITIA59021.2023.10221025](https://doi.org/10.1109/ISITIA59021.2023.10221025).
- [19] B. Xu et al., "Automated cattle counting using Mask R-CNN in quadcopter vision system," *Comput Electron Agric*, vol. 171, Apr. 2020, doi: [10.1016/j.compag.2020.105300](https://doi.org/10.1016/j.compag.2020.105300).
- [20] W. Ni, Z. Song, C. Zhang, and L. Bai, "Improved flame detection based on global attention mechanism and deformable convolution," *Systems and Soft Computing*, vol. 7, Dec. 2025, doi: [10.1016/j.sasc.2025.200406](https://doi.org/10.1016/j.sasc.2025.200406).
- [21] I. Shafi, A. Mazahir, A. Fatima, and I. Ashraf, "Internal defects detection and classification in hollow cylindrical surfaces using single shot detection and MobileNet," *Measurement (Lond)*, vol. 202, Oct. 2022, doi: [10.1016/j.measurement.2022.111836](https://doi.org/10.1016/j.measurement.2022.111836).
- [22] X. Ma et al., "High-throughput maize seed ears sorting through structural re-parameterisation classification model and multi-channel sorting system," *Biosyst Eng*, vol. 254, Jun. 2025, doi: [10.1016/j.biosystemseng.2025.104155](https://doi.org/10.1016/j.biosystemseng.2025.104155).
- [23] C. Pan et al., "Real-time UAV-based wheat lodging detection via edge-accelerated improved Mask-RT-DETR," *Smart Agricultural Technology*, vol. 12, Dec. 2025, doi: [10.1016/j.atech.2025.101509](https://doi.org/10.1016/j.atech.2025.101509).
- [24] D. P. Guragain, B. Shrestha, I. Bajracharya, N. Sharma, D. Ghimire, and S. A. Bhat, "OysterMushNet: A Deep Learning-Based Mobile Application for Real-Time Mushroom Disease Detection and Agronomist Support," *Smart Agricultural Technology*, p. 101735, Dec. 2025, doi: [10.1016/j.atech.2025.101735](https://doi.org/10.1016/j.atech.2025.101735).
- [25] M. F. Almufareh, M. Imran, A. Khan, M. Humayun, and M. Asim, "Automated Brain Tumor Segmentation and Classification in MRI Using YOLO-Based Deep Learning," *IEEE Access*, vol. 12, pp. 16189–16207, 2024, doi: [10.1109/ACCESS.2024.3359418](https://doi.org/10.1109/ACCESS.2024.3359418).
- [26] K. D. Kadam, S. Ahirrao, K. Kotecha, and S. Sahu, "Detection and Localization of Multiple Image Splicing Using MobileNet V1," *IEEE Access*, vol. 9, pp. 162499–162519, 2021, doi: [10.1109/ACCESS.2021.3130342](https://doi.org/10.1109/ACCESS.2021.3130342).
- [27] S. Song, J. Zhu, X. Li, and Q. Huang, "Integrate msRCR and mask r-CnN to recognize underwater creatures on small sample datasets," *IEEE Access*, vol. 8, pp. 172848–172858, 2020, doi: [10.1109/ACCESS.2020.3025617](https://doi.org/10.1109/ACCESS.2020.3025617).
- [28] W. Zhang, Y. Wang, G. Shen, C. Li, M. Li, and Y. Guo, "Tobacco Leaf Segmentation Based on Improved MASK RCNN Algorithm and SAM Model," *IEEE Access*, vol. 11, pp. 103102–103114, 2023, doi: [10.1109/ACCESS.2023.3316364](https://doi.org/10.1109/ACCESS.2023.3316364).
- [29] Q. Zhang, X. Chang, and S. B. Bian, "Vehicle-Damage-Detection Segmentation Algorithm Based on Improved Mask RCNN," *IEEE Access*, vol. 8, pp. 6997–7004, 2020, doi: [10.1109/ACCESS.2020.2964055](https://doi.org/10.1109/ACCESS.2020.2964055).
- [30] K. Amulya, D. K. Deepika, and D. P. Kamakshi, "An optimized hyper parameter-based CNN approach for predicting medicinal or non-medicinal leaves," *Advances in Engineering Software*, vol. 172, Oct. 2022, doi: [10.1016/j.advengsoft.2022.103181](https://doi.org/10.1016/j.advengsoft.2022.103181).
- [31] G. Zhou, M. Jun Peng, and H. Wang, "Enhancing prediction accuracy for LOCA break sizes in nuclear power plants: A hybrid deep learning method with data augmentation and hyperparameter optimization," *Ann Nucl Energy*, vol. 196, Feb. 2024, doi: [10.1016/j.anucene.2023.110208](https://doi.org/10.1016/j.anucene.2023.110208).
- [32] A. Julianto and A. Sunyoto, "A performance evaluation of convolutional neural network architecture for classification of rice leaf disease," *IAES International Journal of Artificial Intelligence*, vol. 10, no. 4, pp. 1069–1078, Dec. 2021, doi: [10.11591/IJAI.V10.I4.PP1069-1078](https://doi.org/10.11591/IJAI.V10.I4.PP1069-1078).
- [33] S. Bensaadi and A. Louchene, "Low-cost convolutional neural network for tomato plant diseases classification," *IAES*

- International Journal of Artificial Intelligence*, vol. 12, no. 1, pp. 162–170, Mar. 2023, [doi: 10.11591/ijai.v12.i1.pp162-170](https://doi.org/10.11591/ijai.v12.i1.pp162-170).
- [34] R. Sreemathy, M. P. Turuk, S. Chaudhary, K. Lavate, A. Ushire, and S. Khurana, “Continuous word level sign language recognition using an expert system based on machine learning,” *International Journal of Cognitive Computing in Engineering*, vol. 4, pp. 170–178, Jun. 2023, [doi: 10.1016/j.ijcce.2023.04.002](https://doi.org/10.1016/j.ijcce.2023.04.002).
- [35] M. S. Beg, M. Y. Ismail, and M. S. U. Miah, “Evaluating the Performance of a Visual Support System for Driving Assistance using a Deep Learning Algorithm,” *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 34, no. 1, pp. 38–50, Mar. 2024, [doi: 10.37934/araset.34.1.3850](https://doi.org/10.37934/araset.34.1.3850).
- [36] N. Dahiya et al., “Hyper-parameter tuned deep learning approach for effective human monkeypox disease detection,” *Sci Rep*, vol. 13, no. 1, Dec. 2023, [doi: 10.1038/s41598-023-43236-1](https://doi.org/10.1038/s41598-023-43236-1).
- [37] U. B. Patayon and R. V. Crisostomo, “Peanut leaf spot disease identification using pre-trained deep convolutional neural network,” *International Journal of Electrical and Computer Engineering*, vol. 12, no. 3, pp. 3005–3012, 2022, [doi: 10.11591/ijeece.v12i3.pp3005-3012](https://doi.org/10.11591/ijeece.v12i3.pp3005-3012).
- [38] A. Victor Ikechukwu, S. Murali, R. Deepu, and R. C. Shivamurthy, “ResNet-50 vs VGG-19 vs training from scratch: A comparative analysis of the segmentation and classification of Pneumonia from chest X-ray images,” *Global Transitions Proceedings*, vol. 2, no. 2, pp. 375–381, Nov. 2021, [doi: 10.1016/j.gltp.2021.08.027](https://doi.org/10.1016/j.gltp.2021.08.027).
- [39] M. Thu, N. Suvonvorn, and N. Kittiphattanabawon, “Pedestrian classification on transfer learning based deep convolutional neural network for partial occlusion handling,” *International Journal of Electrical and Computer Engineering*, vol. 13, no. 3, pp. 2812–2826, Jun. 2023, [doi: 10.11591/ijeece.v13i3.pp2812-2826](https://doi.org/10.11591/ijeece.v13i3.pp2812-2826).
- [40] Y. Aufar and T. P. Kaloka, “Robusta coffee leaf diseases detection based on MobileNetV2 model,” *International Journal of Electrical and Computer Engineering*, vol. 12, no. 6, pp. 6675–6683, Dec. 2022, [doi: 10.11591/ijeece.v12i6.pp6675-6683](https://doi.org/10.11591/ijeece.v12i6.pp6675-6683).
- [41] S. Z. M. Zaki, M. A. Zulkifley, M. Mohd Stofa, N. A. M. Kamari, and N. A. Mohamed, “Classification of tomato leaf diseases using mobilenet v2,” *IAES International Journal of Artificial Intelligence*, vol. 9, no. 2, pp. 290–296, Jun. 2020, [doi: 10.11591/ijai.v9.i2.pp290-296](https://doi.org/10.11591/ijai.v9.i2.pp290-296).
- [42] T. Anwar and S. Zakir, “Vehicle make and model recognition using mixed sample data augmentation techniques,” *IAES International Journal of Artificial Intelligence*, vol. 12, no. 1, pp. 137–145, Mar. 2023, [doi: 10.11591/ijai.v12.i1.pp137-145](https://doi.org/10.11591/ijai.v12.i1.pp137-145).
- [43] Y. S. Devi and S. P. Kumar, “A deep transfer learning approach for identification of diabetic retinopathy using data augmentation,” *IAES International Journal of Artificial Intelligence*, vol. 11, no. 4, pp. 1287–1296, Dec. 2022, [doi: 10.11591/ijai.v11.i4.pp1287-1296](https://doi.org/10.11591/ijai.v11.i4.pp1287-1296).
- [44] N. A. M. Roslan, N. M. Diah, Z. Ibrahim, Y. Munarko, and A. E. Minarno, “Automatic plant recognition using convolutional neural network on Malaysian medicinal herbs: the value of data augmentation,” *International Journal of Advances in Intelligent Informatics*, vol. 9, no. 1, pp. 136–147, Mar. 2023, [doi: 10.26555/ijain.v9i1.1076](https://doi.org/10.26555/ijain.v9i1.1076).
- [45] R. Padilla, S. L. Netto, E. A. B. Da Silva, and S. L. Netto, “A Survey on Performance Metrics for Object-Detection Algorithms Lanterna para Deficientes Visuais (A Device for Helping Blind People) View project A Survey on Performance Metrics for Object-Detection Algorithms”, [doi: 10.1109/IWSSIP48289.2020](https://doi.org/10.1109/IWSSIP48289.2020).
- [46] F. Yang, Y. Zhong, H. Yang, Y. Wan, Z. Hu, and S. Peng, “Microbial Colony Detection Based on Deep Learning,” *Applied Sciences (Switzerland)*, vol. 13, no. 19, Oct. 2023, [doi: 10.3390/app131910568](https://doi.org/10.3390/app131910568).