IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 31, No. 3, March, 2026, pp. 911-920
Journal homepage: http://iieta.org/journals/isi

Optimizing Data Pipeline for Deep Learning Classification of Used Integrated Circuit Components

Iman Fushshilat, Achmad Rizal*, Willy Anugrah Cahyadi

School of Electrical Engineering, Telkom University, Bandung 40287, Indonesia

Corresponding Author Email: achmadrizal@telkomuniversity.ac.id

https://doi.org/10.18280/isi.310322

Received: 12 December 2025
Revised: 18 February 2026
Accepted: 20 March 2026
Available online: 31 March 2026

Keywords:
computer vision, convolutional neural network, data augmentation, deep learning, e-waste, integrated circuit classification, MobileNetV2, transfer learning, automated sorting

ABSTRACT

The rapid rise in electronic device consumption has led to significant e-waste, posing both environmental and economic challenges. In laboratory environments, used Integrated Circuit (IC) components remain functional but are often discarded because of difficulties in manual sorting. This study introduces a deep learning model for the automatic classification of 10 types of used ICs, utilizing a newly collected dataset of 2,016 images. The study addresses challenges related to dataset size and imbalance through systematic data pipeline optimization. The research is conducted in two stages: Stage 1 compares four convolutional neural network (CNN) architectures—custom CNN, EfficientNetB0, EfficientNetB2, and MobileNetV2—to identify the optimal base model. Stage 2 optimizes the data preprocessing pipeline through seven scenarios, incorporating varying data split schemes and compression settings. Key methodologies include YOLOv8-based automatic cropping, SSIM-based adaptive compression, offline data augmentation, and class balancing via upsampling. The results show that pipeline optimization is crucial, improving accuracy from 17% (custom CNN) to 91% (Stage 1 best), and achieving 98.11% classification accuracy in Stage 2 on an independent test set. The optimal configuration was MobileNetV2 with 70:20:10 data split and non-compressed images. While compression reduces storage by 75%, it leads to an 8-9% accuracy drop. This study offers a reproducible methodology for creating high-precision classifiers using minimal data, providing an effective solution for automated IC sorting in laboratory electronic waste management.

1. INTRODUCTION

The use of electrical and electronic equipment (EEE) is closely linked to global economic growth. When you throw away EEE, it creates waste streams (e-waste) that contain dangerous materials that are harmful for the environment and human health. According to the data, there were 53.6 million tons of electronic waste in the world in 2019, which is an average of 7.3 kg per person [1]. E-waste is one of the fastest-increasing types of solid waste, growing by 3% to 5% each year [2]. This rapid increase in electronic waste is detrimental to the environment, human health, and the goals of sustainable development [3].

One solution to the problems caused by e-waste is the implementation of a circular economic approach, in which e-waste is collected, sorted, and processed to extract valuable materials such as gold, copper, and silver. The global monetary value of electronic waste raw materials is estimated to reach 57 billion US dollars, however only 10 billion dollars' worth of electronic waste is recycled and recovered sustainably, with the process producing 15 million tons of CO_2 emissions [4]. Traditionally, innovation in e-waste sorting has relied on electronic sensor technology that detects based on contours [5]. However, this method has significant limitations in recognizing surface patterns and distinguishing between visually similar components. The utilization of computer vision, particularly using ubiquitous camera sensors, has been widely explored for various applications, from wireless data communication [6] to industrial-object classification.

This limitation is particularly acute in the classification of Integrated Circuits (ICs), which present a unique computer vision challenge distinct from other e-waste components. Unlike resistors or capacitors which can often be distinguished by color bands or shape variations, ICs—especially logic gates such as the 74xx series—exhibit extreme inter-class visual similarity. Different functional types (e.g., AND gate IC 7408 vs. NOT gate IC 7404) share identical physical packages (DIP-14), dimensions, and pin configurations (as illustrated in Figure 1). The only distinguishing feature is the alphanumeric code printed on the package surface, which in used components is often faded, scratched, or obscured by oxidation. This high degree of visual ambiguity renders standard contour-based detection insufficient and necessitates a deep learning approach that can extract fine-grained textural features.

Several studies have been conducted to automate the classification of electronic components using machine learning and deep learning approaches [7]. Chand and Lal [8] compared three methods—shallow neural networks (SNNs), support vector machines (SVMs), and deep learning

convolutional neural networks (CNNs)—for component classification. Their research showed the best results using CNN with an accuracy of 98.4% on the dataset and 98.1% in real-world testing. However, this study was limited to only three component types: electrolytic capacitors, potentiometers, and regulators, meaning the model cannot be generalized to other component classes.

Classification efforts for different electronic components have explored several deep learning architectures. Atik [9] achieved 98.99% accuracy in classifying capacitors, resistors, and diodes using a custom CNN, outperforming pretrained networks such as AlexNet (95.42%), GoogleNet (96.67%), ShuffleNet (92.95%), and SqueezeNet (93.86%). The use of hierarchical CNN (NH-CNN) by Hu [10] claimed to reduce computational complexity while achieving 94.26% accuracy. In addition, CNNs, Vision Transformer (ViT) has been explored for electronic component recognition [11], achieving 98.4% accuracy on a dataset of 4,170 images covering 5 component types. Soylu and Kaya [12] compared multiple architectures on 4,038 images, with ResNet achieving the highest accuracy 99.97%.

Despite these advances, a critical gap exists in the literature: there is insufficient research on Integrated Circuit (IC) classification, which represents a crucial category of electronic components commonly used in laboratories and educational settings. Unlike passive components (resistors, capacitors) or simple active components (diodes, transistors), ICs present unique classification challenges owing to their complex packaging, smallprinted text, and high visual similarity between different types.

This study addresses these gaps by developing a comprehensive deep learning classification system specifically for used IC components, with an emphasis on systematic data pipeline optimization. While prior studies have achieved high accuracy on general electronic components [8-12], they have predominantly focused on architectural comparisons (e.g., CNN vs. ViT) rather than data engineering. Crucially, these studies often overlook the systematic optimization of the data pipeline, specifically data splitting strategies, compression effects, and handling of class imbalance, which are determinant factors when dealing with the high inter-class visual similarity of ICs. Consequently, a significant gap remains in the literature regarding comprehensive guidelines for constructing effective classification pipelines for small specialized component datasets.

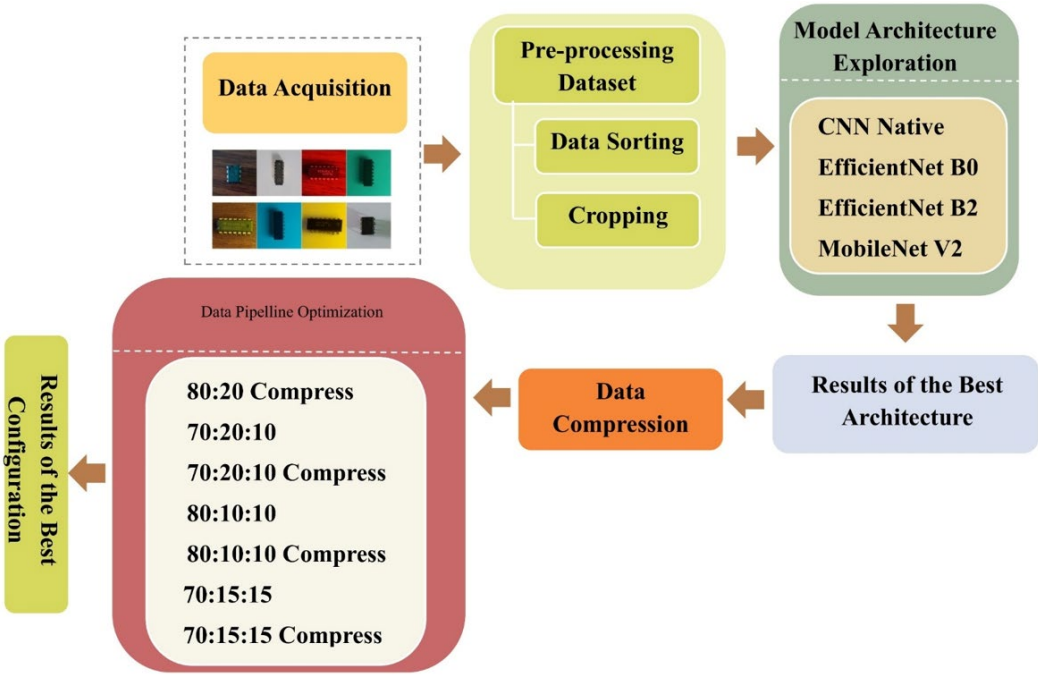

Figure 1. Overall workflow of the component classification process

2. METHODOLOGY

The process of detecting used electronic components in this experiment was performed using a dataset obtained independently. The dataset underwent pre-processing to ensure adequate data quality before being used in the model training, validation, and testing processes. Next, the dataset was processed using various models divided into two stages. These research stages are designed sequentially to determine the optimal classification model configuration.

The process illustrated in Figure 1 outlines the overall workflow, beginning with data acquisition, preprocessing, model development, and evaluation. Each stage ensured that the classification model is systematically refined to achieve optimal performance across all scenarios tested in this study.

2.1 Dataset collection and characteristics

The dataset was constructed specifically for this research and used in our previous study [13], it comprised 2,016 original images across 10 classes of used ICs, captured using a high-resolution camera under controlled laboratory lighting to ensure marking clarity. To introduce realistic variability and prevent background bias, components were photographed on a standard workbench surface with diverse physical configurations, including multiple rotation angles (0°–270°) and varying frame positions (centered to off-centered), thereby simulating the visual context of a practical sorting

environment.

However, as detailed in Table 1, the collected dataset exhibited a significant natural class imbalance. This reflects real-world usage patterns, where common components such the IC Timer 555 (338 images) are far more prevalent than specialized logic gates such the IC 7404 (155 images). To address this imbalance and standardize the input quality, the raw dataset subsequently underwent a systematic preprocessing pipeline.

Table 1. Dataset characteristics

Parameter	Value	
Total Original Images	2,016	
Dataset Distribution	IC Timer 555	:338
	IC OPAMP 324	:157
	IC OPAMP 741	:191
	IC CMOS 4051	:180
	IC TTL 7400	:169
	IC TTL 7402	:214
	IC TTL 7404	:155
	IC TTL 7408	:170
	IC TTL 7432	:265
	IC TTL 7490	:177
Image Resolution (after cropping)	800 × 800 pixels	
Dataset Size (non-compressed)	387 MB	
Dataset Size (compressed)	96.4 MB	
Compression Ratio	75.1%	

2.2 Data preprocessing pipeline

The preprocessing pipeline constitutes the foundational element of this research, meticulously crafted to address three pivotal challenges: the limited size of the dataset, class imbalance, and the optimization of data quality.

2.2.1 Automatic cropping using YOLOv8

To standardize the dataset, automatic object detection and cropping were performed using YOLOv8 [14]. This process ensured that each image contained only the IC component of interest, centered and properly sized with a standardized dimension of 800 × 800 pixels. This approach eliminates background noise and focuses the model's attention on relevant features.

2.2.2 Adaptive compression based on Structural Similarity Index Measure

To evaluate the impact of data quality on model performance in Stage 2, an adaptive compression pipeline was developed using SSIM [15]. Each image was compressed to the lowest JPEG quality level while maintaining an SSIM score of 0.95 or higher. This approach reduced the dataset size by 75.1% (from 387 to 96.4 MB) while preserving structural integrity.

2.2.3 Data splitting

Following the initial preprocessing, the dataset was partitioned into training and validation subsets. Crucially, this splitting was performed before any augmentation to strictly prevent data leakage, ensured that the validation sets remained composed entirely of original, non-synthetic images.

To comprehensively evaluate the robustness of the pipeline, we implemented two distinct splitting strategies corresponding to the experimental stages. Stage 1 used a fixed split ratio (e.g., 80:20) to establish a standard performance benchmark using the raw dataset. Stage 2, to investigate the sensitivity of the optimized pipeline to data availability, we experimented with varying split ratios (Figure 1).

2.3 Upsampling and augmentation strategy

To rectify the significant class imbalance identified in Section 2.1, we implemented an offline augmentation strategy [16] designed to up-sample the minority classes explicitly within the Training Set. Unlike online augmentation which modifies images on-the-fly, offline augmentation physically generates new image files, allowing us to precisely control the class distribution and ensure every class reaches a balanced target of 2,028 images, with total 20,280 images.

The augmentation strategy incorporates two categories of transformations to simulate real-world variability:

(1) Geometric Transformations: To address the "position bias" where components might be placed in various alignments, we applied random rotations (± 20°) and horizontal flips.

(2) Photometric Adjustments: To mitigate the "source bias" caused by uniform laboratory lighting, we introduced brightness and contrast variations (± 20%). This forces the model to learn features that are robust to lighting changes rather than memorizing the specific illumination of the original capture.

2.4 Experimental design and evaluation framework

The research was conducted through two carefully designed experimental stages, each addressing specific research questions regarding model architecture selection and data pipeline optimization.

2.4.1 Stage 1: Model architecture exploration

In the first experimental stage, four different CNN architectures were systematically evaluated to identify the optimal base model for IC classification. The architectures were selected to represent different design philosophies and complexity levels: (1) Custom CNN trained from scratch: a baseline 6-layer CNN with approximately 2.5 million parameters, designed to establish performance expectations without transfer learning; (2) EfficientNetB0 [17]: a compound-scaled CNN optimizing accuracy and efficiency through coordinated scaling of depth, width, and resolution; (3) EfficientNetB2 [18]: a larger variant of EfficientNet with 7.7 million parameters, tested to determine whether increased model capacity benefits this specific task; (4) MobileNetV2 [19]: an efficient architecture using inverted residual blocks and depth-wise separable convolutions, designed for mobile and embedded deployment.

All transfer learning models (EfficientNetB0, EfficientNetB2, MobileNetV2) were initialized using ImageNet pre-trained weights, leveraging features learned from 1.2 million diverse images across 1,000 classes. The custom CNN was trained from scratch with random weight initialization to isolate the contribution of transfer learning. At this stage, a simple 80:20 train-validation split was used without a separate test set, as the primary goal was relative architecture comparison rather than final performance estimation. The preprocessing pipeline consisted of basic cropping and augmentation, intentionally using a non-optimal configuration to stress-test each architecture's ability to handle challenging data conditions.

The performance was evaluated primarily on validation

accuracy, with additional monitoring of training dynamics (loss convergence, overfitting indicators) and computational efficiency (training time, memory usage). This stage aimed to obtain architecture which provides the best balance of accuracy, training efficiency, and generalization capability for IC classification.

2.4.2 Stage 2: Data pipeline optimization

Building on the best-performing architecture from Stage 1 (MobileNetV2), the second stage systematically optimized the data pipeline through a comprehensive experimental design tested seven distinct scenarios.

Seven experimental scenarios were designed to explore these questions: Scenario 1 (80:20 split, compressed data) served as the baseline, using the most common split scheme with compressed images; Scenarios 2-3 (70:20:10 split) introduced a separate 10% test set while maintaining a substantial 20% validation set, comparing non-compressed (Scenario 2) and compressed (Scenario 3) data; Scenarios 4-5 (70:15:15 split) tested equal-sized validation and test sets to examine whether a larger test set improves performance estimation reliability; Scenarios 6-7 (80:10:10 split) allocated maximum data to training while maintaining small but distinct validation and test sets.

Each scenario was evaluated using both validation and test accuracies (except Scenario 1 which lacked a test set). Data split schemes were implemented using stratified sampling to ensure proportional class representation across training, validation, and test sets. All scenarios used the optimal preprocessing pipeline (YOLOv8 cropping, augmentation, and upsampling) except for the compression variable [20]. Training hyperparameters and model architecture remained constant across scenarios to isolate the impact of data split and compression.

This systematic experimental design allows for rigorous comparison and clear identification of optimal configurations. By testing orthogonal factors (split scheme and compression) across multiple scenarios, the study provides comprehensive insights into data pipeline design for small dataset classification tasks.

2.5 Model architecture and training strategy

To ensure a fair assessment of architectures with vastly different capacities, we adopted an adaptive hyperparameter configuration instead of a rigid protocol. We tailored the training strategy to the convergence behavior of each model, where high-capacity models used fine-tuning and grayscale preprocessing to mitigate overfitting. Conversely, lightweight models such as MobileNetV2 achieved optimal stability using a standard feature extraction mode with frozen weights. This approach isolates the maximum potential of each architecture as detailed in Table 2.

Table 2. Architecture-specific configurations

Feature	Custom CNN	Efficient NetB0	Efficient NetB2	Mobile NetV2
Param. Count	2.5 M	4.0 M	7.7 M	**2.3 M**
Training Mode	From Scratch	Frozen (Feat. Ext.)	Fine-Tuning (2-Phase)*	**Frozen (Feat. Ext.)**
Pre-processing	RGB	RGB	**Grayscale**	RGB
Key Config.	Adam, 50 Epochs, LR 1e-3	Cosine Decay, 75 Epochs	2-Phase (30+30 eps), LR 1e-5	Adam, 50 Epochs, LR 1e-3

2.6 Model performance and evaluation

The model performance was evaluated using accuracy, precision, recall, and F1-score for each class. Confusion matrices were analyzed to identify problematic class pairs. The model performance was evaluated using four standard metrics derived from the confusion matrix: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). Accuracy (Eq. (1)) measures the overall ratio of correct predictions across all classes. Precision (Eq. (2)) assesses the model's exactness by calculating the proportion of true positive predictions among all positive predictions, while Recall (Eq. (3)) evaluates the completeness by measuring the ability to identify all relevant instances. Finally, the F1-Score (Eq. (4)) provides the harmonic mean of precision and recall, offering a balanced metric crucial for assessing performance on imbalanced datasets [21].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

3. RESULTS AND DISCUSSION

3.1 Stage 1: Architecture exploration and the importance of transfer learning

The experimental design of Stage 1 was structured as a preliminary screening phase to identify the most promising backbone architecture. During this phase, we conducted extensive hyperparameter tuning and exploration experiments on each candidate model. The results reported in this section represent the best-performing model instance (best checkpoint) obtained after tuning. Consequently, performance was evaluated on a fixed validation split to determine the upper bound capability of each architecture, rather than averaging multiple runs which might include suboptimal tuning configurations.

This stage revealed profound differences in model performance across the four tested architectures, providing critical insights into the necessity of transfer learning for small-dataset classification tasks. Figure 2 presents the validation accuracy comparison, dramatically illustrating the performance gap between architectures.

The custom CNN trained from scratch failed catastrophically, achieving only 17% accuracy on the validation set. A detailed analysis of the confusion matrix revealed that the model collapsed into a degenerate solution, predicting only the majority class (IC 555 Timer) for nearly all test samples. This behavior is characteristic of severe underfitting where the model fails to learn meaningful features

and instead adopts the statistically safest strategy of always predicting the most common class. The learning curves showed minimal improvement beyond the first few epochs, with training accuracy plateauing approximately 20% and validation accuracy remaining flat. This complete failure demonstrates unequivocally that the IC classification task is too complex and the dataset too small for effective training from scratch, even with extensive data augmentation.

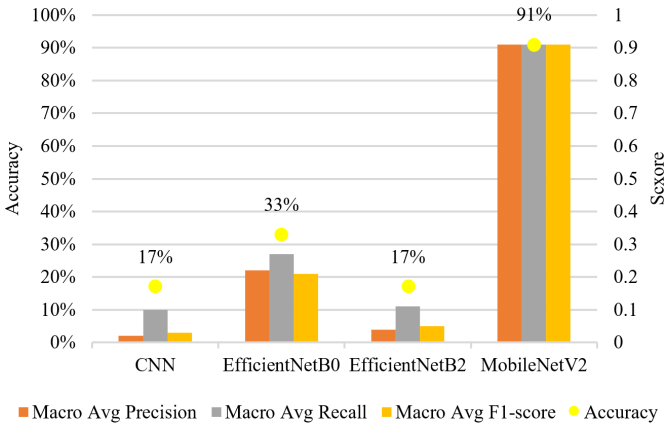

Figure 2. Stage 1 model performance

In stark contrast, EfficientNetB0 with ImageNet transfer learning showed significant improvement, reaching 33% validation accuracy. Although still far from acceptable performance, this represents a doubling of accuracy compared to the from-scratch baseline, powerfully confirming that pre-trained features from ImageNet provide a strong foundation even for specialized industrial classification tasks. The model successfully learned to differentiate between some IC classes, particularly those with distinct physical characteristics (IC OPAMP 324 with 14 pins versus IC 555 with eight pins). However, the learning curves exhibited high volatility in validation metrics, with validation accuracy fluctuating between 28% and 38% across epochs. This instability suggests two issues: (1) the validation set size (403 images) may be insufficient for stable performance estimation, and (2) the data pipeline optimization is suboptimal, as hypothesized for Stage 1 experiments.

Surprisingly, EfficientNetB2, despite being a more powerful architecture with 7.7 million parameters (nearly 2 × more than EfficientNetB0), performed worse than its smaller sibling, achieving a validation accuracy of only 16.83 %. This counterintuitive result provides a crucial lesson regarding model capacity and dataset size. Analysis of training dynamics revealed that EfficientNetB2 exhibited extreme instability from the very beginning, with validation loss increasing while training loss decreased, a classic sign of severe overfitting. The model appeared to memorize training examples almost perfectly (training accuracy >90% within five epochs) while completely failing to generalize to validation data. The excessive capacity of EfficientNetB2, beneficial for large datasets, becomes a liability for small datasets where the model has sufficient parameters to memorize every training example including noise and outliers, leaving no capacity for learning generalizable features. This phenomenon, sometimes called the "curse of complexity," demonstrates that more parameters do not automatically improve performance and can actively harm it in data-limited scenarios.

MobileNetV2 emerged as the clear winner of Stage 1, achieving an impressive 91% validation accuracy. This result is particularly noteworthy given that MobileNetV2 has only 2.3 million parameters, fewer than even the custom CNN (2.5 million) and less than half of EfficientNetB2 (7.7 million) [22, 23].

The success of MobileNetV2 can be attributed to three factors: (1) Architectural efficiency: MobileNetV2's depth-wise separable convolutions and inverted residual blocks with linear bottlenecks provide an optimal balance of representation capacity and parameter efficiency, reducing overfitting risk; (2) Appropriate capacity: With moderate size, MobileNetV2 has sufficient capacity to learn IC-specific features without the overfitting problems of EfficientNetB2; (3) Effective transfer learning: The ImageNet pre-trained features transfer well to IC classification, capturing low-level edge and texture patterns crucial for distinguishing printed text and surface markings on ICs.

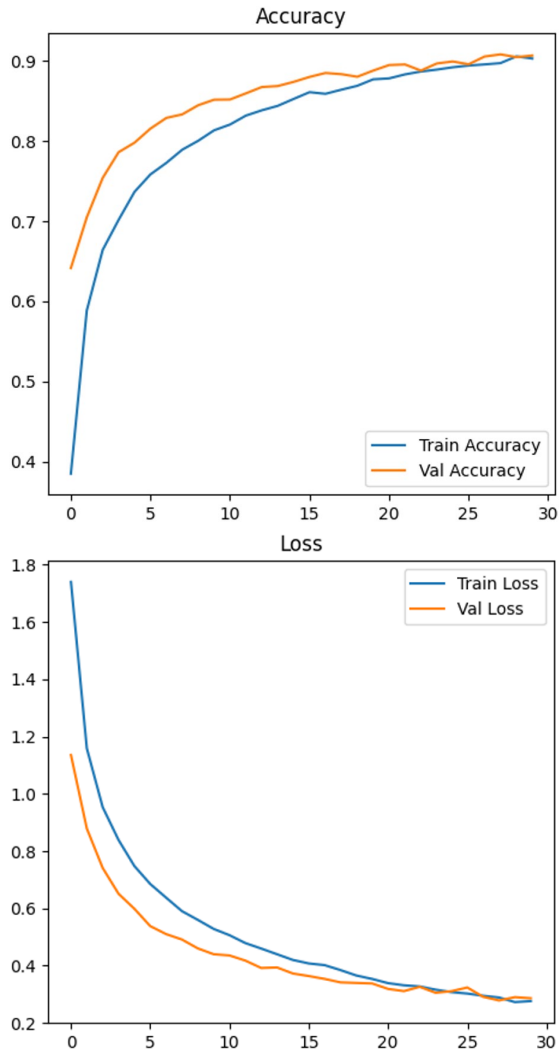

Figure 3. MobileNetV2's learning curves

Analysis of MobileNetV2's learning curves (Figure 3) revealed healthy convergence with training and validation accuracy curves tracking closely throughout training (final gap < 5%), indicating good generalization without significant overfitting. The model converged smoothly within 20 epochs, achieving stable validation accuracy above 90% by epoch 15.

Per-class analysis showed strong performance across most IC types (F1-scores >0.85 for eight out of ten classes), with difficulty primarily in two classes: IC 7404 and IC 7490, which are visually very similar to other ICs in the 74xx family

(differing only in small-printed numbers). This pattern of confusion between visually similar ICs persist even with pipeline optimization, suggesting fundamental perceptual challenges rather than model limitations.

The Stage 1 results conclusively established MobileNetV2 as the optimal architecture for IC classification and provided compelling evidence for the critical importance of transfer learning in small-dataset domains. The confusion matrix of MobileNetV2 performance is shown in Figure 4. The 74-percentage point improvement from custom CNN (17%) to MobileNetV2 (91%) represents the largest single factor affecting model performance, dwarfing potential gains from hyperparameter tuning or architectural tweaks. However, the Stage 1 experiments also revealed room for improvement through data pipeline optimization, motivating the systematic investigation in Stage 2.

3.2 Stage 2: Pipeline ablation study and performance optimization

Building on MobileNetV2 as the optimal architecture, Stage 2 serves as a systematic ablation study to quantify the contribution of each pipeline component. By varying the data split ratios and compression settings, we isolate the specific impact of data engineering strategies on model performance. The results, presented comprehensively in Table 3, reveal distinct performance hierarchies across different configurations.

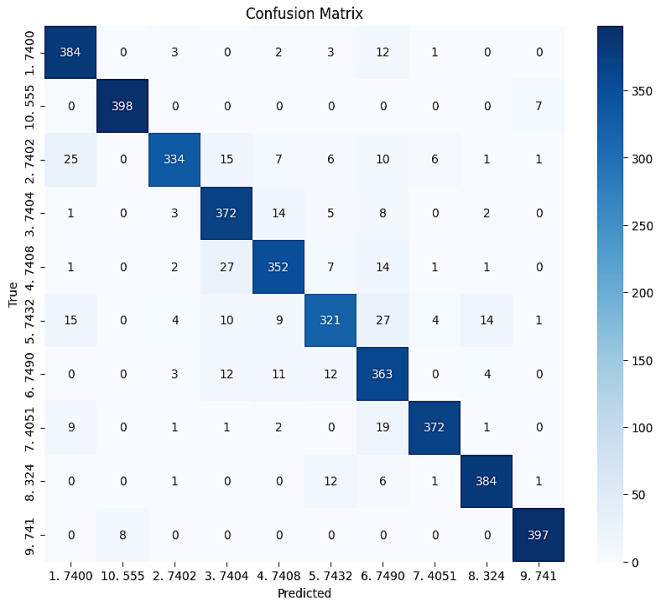

Figure 4. MobileNetV2 confusion matrix

Table 3. Ablation study of data pipeline configurations (split scheme & compression)

Experiment	Architecture	Dataset	Spliting Scheme	Augmentasi Offline	Upsampling	Accuracy Val (%)	Accuracy Test (%)
Stage 1	MobileNet V2	Non- Compression	80:20:00	No	No	49	-
	MobileNet V2 (Best)	Non- Compression	80:20:00	Yes	Yes	91	-
		Compression	80:20:00	Yes	Yes	88	-
Stage 2	MobileNet V2	Non- Compression	70:20:10	Yes	Yes	95	98
		Compression	70:20:10	Yes	Yes	87	89
		Non- Compression	70:15:15	Yes	Yes	91	90
		Compression	70:15:15	Yes	Yes	85	89
		Non- Compression	80:10:10	Yes	Yes	89	93
		Compression	80:10:10	Yes	Yes	92	91

3.2.1 Impact of data split scheme on model performance

The choice of data-split scheme proved to have a substantial and nuanced impact on model performance and evaluation reliability. The 70:20:10 split (70% training, 20% validation, 10% test) consistently produced the most stable and highest-performing models for both compression settings. Scenario 2, using this split with non-compressed data, achieved a remarkable 98% test accuracy, representing a 7-percentage point improvement over the best Stage 1 result (91%) and an 81-percentage point improvement over the from-scratch baseline (17%). This dramatic improvement, achieved using the same model architecture as Stage 1 but with an optimized data pipeline, demonstrates that data engineering is the dominant factor in small-dataset classification performance.

The success of the 70:20:10 split can be explained by multiple mechanisms. First, the substantial validation set (20% of data, or 403 images) provides reliable performance estimation for model selection via early stopping and ModelCheckpoint callbacks. With 40 images per class on average, the validation set contains sufficient samples to capture intra-class variability and detect overfitting. Second, despite allocating only 70% to training (compared to 80% in Scenario 1), the extensive augmentation and upsampling strategies expand the effective training set size by 6× through generated variants, more than compensating for the smaller base training set. Third, the separate 10% test set (202 images) provides unbiased final performance evaluation on completely unseen data, which is lacking in Stage 1 experiments.

In contrast, the 70:15:15 split (equal validation and test sets) performed notably worse, with Scenario 4 achieving only 90% test accuracy despite using non-compressed data—an 8-percentage point drop compared to Scenario 2. The analysis revealed that a smaller validation set (302 images, or ~30 images per class) led to more volatile validation metrics during training. The validation loss fluctuated significantly across epochs, causing early stopping to trigger prematurely in some runs or too late in others, resulted in suboptimal model selection. This demonstrates that validation set size has a threshold effect: below approximately 35-40 images per class, the set becomes too small for reliable performance monitoring.

The 80:10:10 split, which allocated maximum data to training, achieved respectable performance (Scenario 6: 93% test accuracy with non-compressed data) but fell short of the 70:20:10 split. While the larger training set (1,613 images)

initially seems advantageous, two factors limit its benefit: (1) the small validation set (202 images, equal to the test set) suffers from the same reliability issues as the 70:15:15 split, leading to less effective model selection; (2) the marginal benefit of additional training samples diminishes when extensive augmentation is already applied—the difference between 1,411 and 1,613 base training images becomes negligible when each generates 6 augmented variants. These results suggest that beyond a certain training set size (approximately 1,400 images for this task), investing additional data in a larger, more reliable validation set provides greater performance benefits than further expanding the training set.

3.2.2 Impact of data compression

The Accuracy-Efficiency Trade-off - Compression exhibited a remarkably consistent detrimental effect on model performance, reducing test accuracy by approximately 8-9 percentage points across all three data split schemes tested, as illustrated in Figure 5. This consistency suggests that performance degradation is intrinsic to the compression process rather than an interaction effect with data splitting. Specifically, Scenario 3 (70:20:10 compressed) achieved 89% test accuracy compared to 98% for Scenario 2 (non-compressed), Scenario 5 (70:15:15 compressed) achieved 89% versus 90% non-compressed, and Scenario 7 (80:10:10 compressed) achieved 91% versus 93% non-compressed.

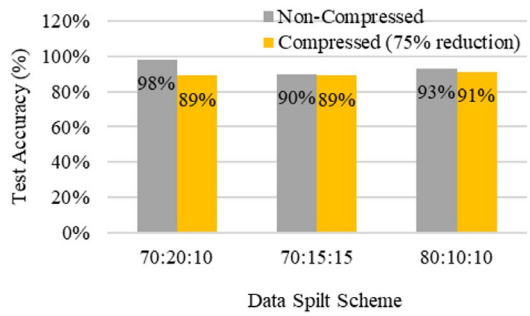

Figure 5. Impact of data compression on model performance across different data splits

A detailed analysis of per-class performance revealed that compression had non-uniform effects across IC types. The most severe accuracy drops occurred for visually similar ICs in the 74xx family, particularly IC 7404, IC 7408, IC 7402, and IC 7490. For these classes, F1-scores dropped by 10-15 percentage points with compression. This pattern makes intuitive sense: these ICs are nearly identical in physical appearance, differing only in small-printed numbers (e.g., "7404" vs "7408"). The subtle differences in text rendering—fine serif details, slight variations in font weight, minor edge sharpness—are exactly the high-frequency spatial information preferentially removed by JPEG compression.

In contrast, ICs with distinctive physical characteristics (IC OPAMP 324 with its quad package, IC 555 with unique pin configuration, IC CMOS 4051 with its multiplexer marking) showed minimal accuracy degradation with compression (F1-score drops < 3%). These ICs can be classified based on gross structural features, package shape, pin count, large-printed logos—which are preserved even at moderate compression levels. This finding has important practical implications: for IC classification systems, compression may be acceptable if the IC types are physically distinctive, however high-resolution, non-compressed images become essential when distinguishing between closely related IC families.

The SSIM-based adaptive compression achieved an impressive 75.1% reduction in dataset size (387 MB → 96.4 MB), representing substantial savings in storage and transmission bandwidth. For deployment in resource-constrained environments (edge devices, mobile applications, bandwidth-limited networks), this storage reduction could be critical. The results quantify a clear accuracy-efficiency trade-off: a 75% storage reduction costs approximately 9% accuracy. Whether this trade-off is acceptable depends on application requirements. For laboratory sorting where accuracy is paramount and storage is cheap, non-compressed images are preferable. For mobile sorting applications or large-scale deployment where storage costs dominate, the compressed model delivering 89-91% accuracy may be more practical than a 98% accurate model requiring four times more storage.

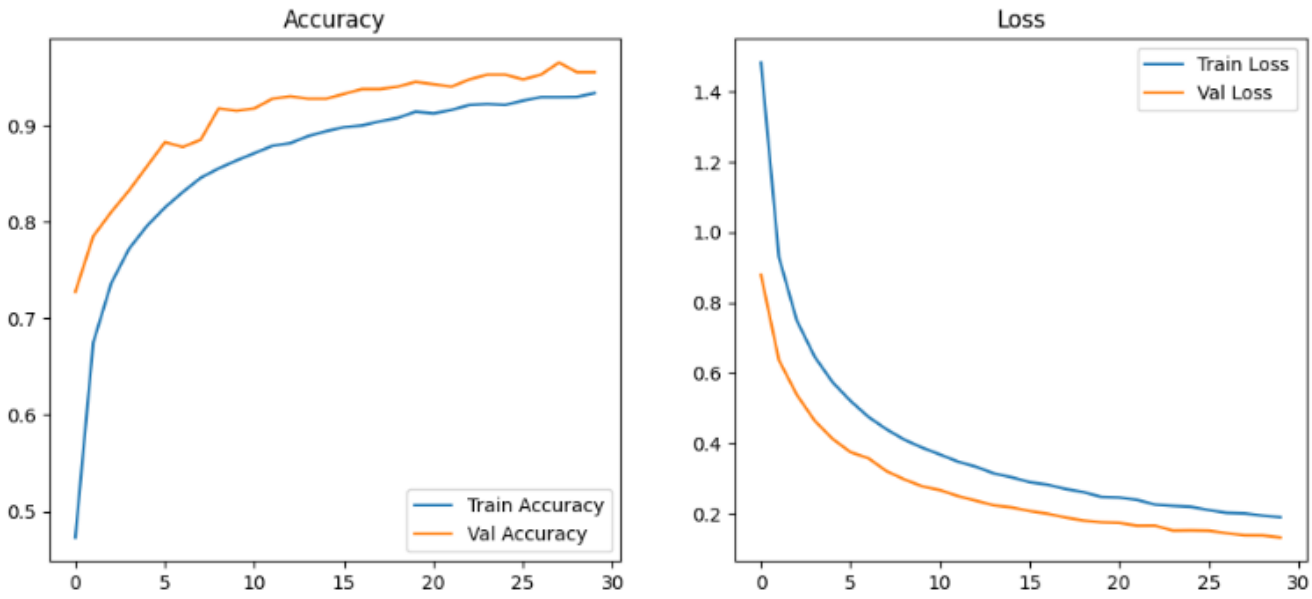

Figure 6. Training and validation accuracy and loss curves for the best model

Interestingly, Scenario 7 (80:10:10 compressed, 91% accuracy) outperformed all other compressed scenarios despite using the same compression. This demonstrates that an increase in the quantity of training data can partially compensate for the reduction in data quality. With 1,613 training images (versus 1,411 in Scenarios 2-5), the model had more examples to learn robust features despite compression artifacts. This finding suggests a potential optimization strategy: when compression is necessary, should be maximized training set size to mitigate accuracy degradation. However, even with maximum training data, compressed models (91%) still fall 7 percentage points short of the optimal non-compressed configuration (98%), indicating fundamental information loss that cannot be fully compensated by quantity alone.

3.2.3 Learning dynamics of the optimal model

Figure 6 presents the learning curves for the best-performing model (Scenario 2: MobileNetV2, 70:20:10 split, non-compressed data), providing insights into the training process and generalization behavior.

The learning curves exhibit several noteworthy characteristics that validate the training strategy. First, there was smooth, monotonic improvement in both training and validation accuracy throughout Phase 1 (epochs 1-15), with validation accuracy rising from 42% to 95% and close tracking training accuracy (gap <3%). This tight coupling indicates that the model learns generalizable features rather than memorizing training data, a crucial property for small-dataset scenarios. The effectiveness of dropout (rate=0.3) and data augmentation in preventing overfitting is clearly demonstrated.

Second, the transition to Phase 2 (fine-tuning) at epoch 15 is marked by continued improvement but at a slower rate, with validation accuracy incrementally increasing from 95% to 95% (plateau around epoch 20). The dramatically reduced learning rate (1×10^{-5}) allows subtle refinements to the pre-trained weights without destabilizing the learned features. Importantly, the validation loss continues to decrease during fine-tuning without increasing the training-validation gap, confirming that fine-tuning genuinely improvs the model rather than overfitting. The early stopping mechanism correctly identified the optimal model at epoch 22, when validation loss reached its minimum.

Third, the absolute loss values provided confidence in model calibration. The final validation loss of approximately 0.18 corresponds well with the 95% validation accuracy, and the loss curves show no signs of divergence or instability. Stable, low-loss values indicate that the model produces confident, well-calibrated predictions rather than barely crossing the decision boundary. This is important for practical deployment where prediction confidence scores inform system behavior (e.g., flagging uncertain predictions for manual review).

3.2.4 Per-class performance analysis

The confusion matrix in Figure 7 provides granular insights into the classification performance across all 10 IC types, revealing both the strengths and challenges of the optimal model.

The confusion matrix demonstrates overwhelming dominance of the diagonal, with 207 out of 211 test samples (98.11%) correctly classified. Only five misclassifications occurred across the entire test set, all involving ICs within the 74xx logic gate family: IC 7404 was misclassified as IC 7408 once (6.25% of IC 7404 samples), IC 7402 was misclassified as IC 7490 once (4.35% of IC 7402 samples), IC 7432 was misclassified as IC 7490 once (3.70% of IC 7432 samples), IC 7490 was misclassified as IC 7432 once (5.26% of IC 7490 samples), IC 4051 was misclassified as IC 7490 once (5.26% of IC 4051 samples).

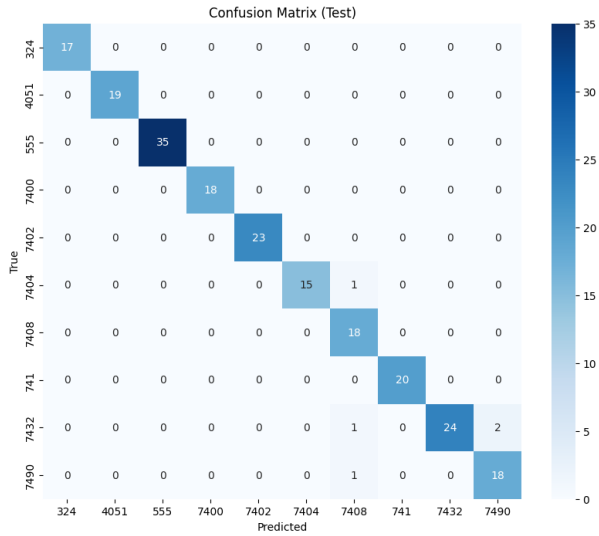

Figure 7. Confusion matrix of the best model on test data

This error pattern was highly informative. All misclassifications involved either ICs within the 74xx series or confusion between 74xx and other DIP-packaged ICs. This is expected given that 74xx series ICs are visually nearly identical—same package size, same pin count, same labeling format—with differences only in the specific printed numbers. Even human experts sometimes require careful examination to distinguish "7404" from "7408" on a worn or poorly lit IC. The model's 98% accuracy despite these fundamental perceptual challenges is remarkable.

Notably, certain IC classes achieved perfect classification: IC OPAMP 324 (17/17 correct, 100%), IC Timer 555 (35/35 correct, 100%), IC TTL 7400 (18/18 correct, 100%), IC OPAMP 741 (20/20 correct, 100%). These ICs have distinctive physical characteristics and well represented in the training set, allowing the model to learn robust discriminative features. The lone IC 7408 misclassification (18/18 correct when predicted as 7408 but caused one IC 7404 to be incorrectly predicted as 7408) shows that even challenging classes can be learned effectively with the optimized pipeline.

The per-class F1 scores ranged from 0.91 (IC 7408) to 1.00 (IC 324, 555, 7400, 741), with a macro-average F1-score of 0.98. The consistency of high F1-scores across all classes, despite significant class imbalance in the original dataset, validates the effectiveness of the upsampling strategy. Without balancing, minority classes such as IC 741 and IC 7404 would likely show much lower recall, as the model would be biased towards predicting majority classes. The balanced F1-scores confirm that the model learned to distinguish all IC types equally well, regardless of their prevalence in the training data.

3.2.5 Statistical robustness and metric evaluation

Beyond per-class analysis, we prioritized Macro-averaged F1-score to rigorously evaluate global performance on this imbalanced dataset. The proposed model achieved a Macro F1-score of 0.98, closely matching the Test Accuracy of 98%.

This minimal divergence confirms that the model's performance is unbiased and not skewed by majority classes. Regarding statistical variability, although we report the performance of the best model checkpoint without multi-run standard deviations owing to computational constraints, the high consistency between the validation (95%) and test (98%) scores serves as a strong proxy for stability. Furthermore, the magnitude of improvement over the baseline (spanning >80 percentage points) is sufficiently large to establish the proposed method's superiority without requiring tight confidence intervals.

4. CONCLUSIONS

In this study, a highly accurate deep learning system was developed for the automatic classification of 10 types of used IC components, achieving 98% test accuracy through systematic optimization of the data preprocessing pipeline. Based on the extensive experimental analysis, we delineated our findings into generalizable methodological contributions and dataset-specific insights.

4.1 Generalizable methodological contributions

The primary finding was that data pipeline optimization is the dominant factor determining model performance in small-dataset regimes, far exceeding the impact of model architecture selection. The systematic progression from 17% accuracy (custom CNN trained from scratch) to 91% (transfer learning) and finally to 98% (optimized pipeline) clearly demonstrates an 81-percentage point total improvement, transforming the system from non-functional to production-ready. This underscores a critical insight for practitioners: in small-data scenarios, investing effort in data quality, specifically the optimal combination of YOLOv8-based cropping, augmentation-based upsampling, and stratified splitting, yields greater returns than extensive hyperparameter tuning. Furthermore, this rigorous methodology provides a reproducible template can be adapted to other domains with limited samples, such as medical image analysis or industrial defect detection.

4.2 Dataset-specific findings

In the specific context of IC classification, MobileNetV2 emerged as the optimal architecture, outperforming both smaller (EfficientNetB0) and larger (EfficientNetB2) alternatives. The success of MobileNetV2 (2.3 million parameters) over the much larger EfficientNetB2 (7.7 million parameters) demonstrates that moderate model capacity paired with efficient design outweighs raw parameter count for components with limited feature variability. Regarding storage optimization, we quantified a critical trade-off: while SSIM-guided compression achieved a 75% reduction in storage (387 MB to 96.4 MB), it caused a significant 9% drop in accuracy (from 98% to 89%), particularly affecting visually similar IC classes. Thus, for high-accuracy laboratory sorting, uncompressed input remains the optimal configuration.

4.3 Practical impact

The research provides an immediately deployable solution for automated IC sorting in laboratory environments. With 98% accuracy, the system can handle the vast majority of sorting tasks autonomously, enabling systematic component reuse programs in educational institutions. The decision to deploy the compressed model (89-91% accuracy) versus the full model (98% accuracy) depends on specific operational constraints regarding storage bandwidth versus manual review capacity.

4.4 Limitations and future work

We acknowledge several limitations of the current study that present opportunities for further research. First, as the dataset is restricted to through-hole (DIP) packages, expanding the analysis to Surface-Mount Devices (SMD) and real-world damaged ICs would significantly enhance the robustness of the proposed method in modern manufacturing. Second, since this study focused on input-data optimization, investigating model-centric optimization techniques (such as pruning and quantization) remains a promising avenue to further bridging the gap between model size and accuracy. Finally, integrating YOLO-based object detection to enable an end-to-end "detect-and-classify" pipeline would be a valuable extension for real-world bin sorting scenarios.

ACKNOWLEDGEMENT

The authors acknowledge the institutional support provided by the School of Electrical Engineering, Telkom University (Tel-U), Bandung, Indonesia. This work would not have been possible without their generous commitment to fostering advanced research.

REFERENCES

[1] Forti, V., Baldé, C.P., Kuehr, R., Bel, G. (2020). The global e-waste monitor 2020. https://ewastemonitor.info/wp-content/uploads/2020/11/GEM_2020_def_july1_low.pdf.

[2] Liu, K., Tan, Q., Yu, J., Wang, M. (2023). A global perspective on e-waste recycling. Circular Economy, 2(1): 100028. https://doi.org/10.1016/j.cec.2023.100028

[3] Madkhali, H., Duraib, S., Nguyen, L., Prasad, M., Sharma, M., Joshi, S. (2023). A comprehensive review on e-waste management strategies and prediction methods: A Saudi Arabia perspective. Knowledge, 3(2): 163-179. https://doi.org/10.3390/knowledge3020012

[4] Shahabuddin, M., Uddin, M.N., Chowdhury, J.I., Ahmed, S.F., Uddin, M.N., Mofijur, M., Uddin, M.A. (2023). A review of the recent development, challenges, and opportunities of electronic waste (e-waste). International Journal of Environmental Science and Technology, 20(4): 4513-4520. https://doi.org/10.1007/s13762-022-04274-w

[5] Bhoi, N.K. (2024). Advancements in e-waste recycling technologies: A comprehensive overview of strategies and mechatronics integration for future development. Sustainable Materials and Technologies, 42: e01182. https://doi.org/10.1016/j.susmat.2024.e01182

[6] Cahyadi, W.A., Chung, Y.H., Ghassemlooy, Z., Hassan, N.B. (2020). Optical camera communications: Principles, modulations, potential and challenges.

Electronics, 9(9): 1339. https://doi.org/10.3390/electronics9091339
[7] Dong, S., Wang, P., Abbas, K. (2021). A survey on deep learning and its applications. Computer Science Review, 40: 100379. https://doi.org/10.1016/j.cosrev.2021.100379
[8] Chand, P., Lal, S. (2022). Vision-based detection and classification of used electronic parts. Sensors, 22(23): 9079. https://doi.org/10.3390/s22239079
[9] Atik, I. (2022). Classification of electronic components based on convolutional neural network architecture. Energies, 15(7): 2347. https://doi.org/10.3390/en15072347
[10] Hu, X., Xu, J., Wu, J. (2020). A novel electronic component classification algorithm based on hierarchical convolution neural network. IOP Conference Series: Earth and Environmental Science, 474(5): 052081. https://doi.org/10.1088/1755-1315/474/5/052081
[11] Chen, Y., Chen, X. (2023). Image classification of electronic components based on Vision Transformer. In 2023 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML), pp. 407-410. https://doi.org/10.1109/icicml60161.2023.10424884
[12] Soylu, E., Kaya, I. (2024). Classification of electronics components using deep learning. Sakarya University Journal of Computer and Information Sciences, 7(1): 36-45. https://doi.org/10.35377/saucis...1391636
[13] Fushshilat, I., Rizal, A., Cahyadi, W.A. (2025). An optimized deep learning pipeline for used IC classification integrating DenseNet121 with fine-tuning. In 2025 5th International Conference of Science and Information Technology in Smart Administration (ICSINTESA), Yogyakarta, Indonesia, pp. 311-316. https://doi.org/10.1109/ICSINTESA68165.2025.11413724
[14] Du, B., Wan, F., Lei, G., Xu, L., Xu, C., Xiong, Y. (2023). YOLO-MBBi: PCB surface defect detection method based on enhanced YOLOv5. Electronics, 12(13): 2821. https://doi.org/10.3390/electronics12132821
[15] Bakurov, I., Buzzelli, M., Schettini, R., Castelli, M., Vanneschi, L. (2022). Structural similarity index (SSIM) revisited: A data-driven approach. Expert Systems with Applications, 189: 116087. https://doi.org/10.1016/j.eswa.2021.116087
[16] Peng, H., Xu, H., Shen, G., Liu, H., Guan, X., Li, M. (2024). A lightweight crop pest classification method based on improved MobileNet-V2 model. Agronomy, 14(6): 1334. https://doi.org/10.3390/agronomy14061334
[17] Ali, H., Shifa, N., Benlamri, R., Farooque, A.A., Yaqub, R. (2025). A fine tuned EfficientNet-B0 convolutional neural network for accurate and efficient classification of apple leaf diseases. Scientific Reports, 15(1). https://doi.org/10.1038/s41598-025-04479-2
[18] El-Aziz, A.A.A., Elmogy, M., El-Ghany, S.A. (2025). A robust tuned EfficientNet-B2 using dynamic learning for predicting different grades of brain cancer. Egyptian Informatics Journal, 30: 100694. https://doi.org/10.1016/j.eij.2025.100694
[19] Alfarizi, S., Salsabila, Z., Bahrullah, S., Susanti, H., Cahyadi, W. A. (2023). Real-time cat detection system using MobileNet-SSD V2. In 2023 8th International Conference on Instrumentation, Control, and Automation (ICA), Jakarta, Indonesia, pp. 276-280. https://doi.org/10.1109/ica58538.2023.10273134
[20] Wu, D., Liu, Z. (2022). Defect detection method of electronic components based on improved YOLOX. In 2022 International Conference on Algorithms, Data Mining, and Information Technology (ADMIT), Xi'an, China, pp. 56-61. https://doi.org/10.1109/admit57209.2022.00018
[21] Suryani, V., Yulianto, F. A., Sukarno, P., Rizal, A. (2024). A comparison of deep learning and machine learning approaches to video injection detection. Mathematical Modelling of Engineering Problems, 11(12): 3431-3439. https://doi.org/10.18280/mmep.111221
[22] Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., Parmar, M. (2024). A review of convolutional neural networks in computer vision. Artificial Intelligence Review, 57(4): 99. https://doi.org/10.1007/s10462-024-10721-6
[23] Edara, G.R., Reddy, A.R. (2025). Interpretable deep learning framework for early classification of tomato and grapevine leaf diseases. Acadlore Transactions on AI and Machine Learning, 4(3): 174-185. https://doi.org/10.56578/ataiml040303