Improving object detection using a two-stage machine vision and object classification system for smart edu station

Yuliadi Erdani [a,1,*], Abdur Rohman Harits Martawireja [a,2], Rangga Rizky Radityo [a,3], Sarosa Castrena Abadi [a,4], Abyanuddin Salam [b,5], Rizqi Aji Pratama [a,6]

[a] Department of Automation Engineering, Politeknik Manufaktur Bandung, Jl. Kanayakan No.21, Bandung, 40135, West Java, Indonesia
[b] School of Science Computing Emerging Technologies, Swinburne University of Technology, Australia
[1] yul_erdani@yahoo.com; [2] harits@polman-bandung.ac.id; [3] rangga.radityo@gmail.com; [4] sarosa@ae.polman-bandung.ac.id; [5] asalam@swin.edu.au; [6] rizqi@ae.polman-bandung.ac.id
* corresponding author

ARTICLE INFO

Article history
Received April 22, 2026
Revised June 21, 2026
Accepted June 23, 2026
Available online November 5, 2026

Keywords
Machine Vision
Convolutional Neural Network
ROI Localization
Object Classification
Smart Edu Station

ABSTRACT

This paper presents a two–stage machine vision and deep learning–based object classification system capable of recognizing 27 combinatorial classes (shape, color, and size) in real time on the Smart Edu Station platform. Unlike end-to-end object detection frameworks such as YOLO that jointly localize and classify objects in a single pass, this system explicitly decouples the two tasks. The first stage employs HSV (Hue Saturation Value) –based morphological segmentation to localize the Region of Interest (ROI), whereas the second stage delegates feature extraction to a custom Convolutional Neural Network (CNN) architecture built entirely from scratch. The model achieved a validation accuracy of 95.00% with a macro F1-score of 95.00% on 649 validation images, and a test accuracy of 93.00% with a macro F1-score of 92.00% on 355 independent test images. To address temporal instability, the system adopts a Consecutive Stable Detection Window (CSDW) mechanism: a label is declared valid only when the same class is detected consistently for a full 3–second window without interruption; any class change resets the counter to zero. Validated labels are transmitted via TCP/IP encapsulated in JSON format. Under controlled laboratory conditions, testing across three sessions (82 objects in total) yielded a 100% transmission success rate with zero packet loss over an isolated LAN connection.

© 2026 The Author(s).
This is an open access article under the CC–BY-SA license.

1. Introduction

The growing demand for automation within smart manufacturing ecosystems requires conveyor visual inspection systems to classify spatio–temporal object features instantly and robustly [1]. Single–stage object detection architectures such as YOLO have become the industry standard owing to their ability to perform localization and classification simultaneously [2]. Subsequent YOLO iterations have continued to refine this single-stage detection paradigm for improved speed–accuracy trade-offs [3], [4], [5], [6], [7], [8]. However, the high matrix multiplication intensity of these architectures induces extreme computational bottlenecks when deployed on edge computing devices without a GPU, directly degrading frame rate (<15 FPS) and inference latency (>50 ms) [9], [10], [11], [12], [13], [14], [15], as summarized in Table 1.

Attempts to reduce computational load through transfer learning from pre-trained models such as MobileNetV2 [16] or EfficientNet [17] still result in over-parameterized networks that are inefficient for classifying simple 27-class spatial geometry [18]. This study addresses these deficiencies by proposing a two-stage machine vision hybridization [19]: HSV morphological segmentation efficiently extracts the ROI, followed by inference using a custom CNN built from scratch to fundamentally reduce parameter redundancy.

Table 1. Comparison of Object Recognition Architecture Performance Metrics.

Model	Computational Requirement	Frame Rate (FPS)	Inference Latency (ms)	Reference
Single-Stage (YOLOv7/YOLOv8)	High (GPU required)	12-15 FPS	> 50 ms	[2], [9], [10]
Transfer Learning (MobileNetV2)	Medium (Over-parameterized)	18-22 FPS	40-45 ms	[16], [17]
Proposed Two-Stage Hybrid	**Low (CPU only)**	**> 30 FPS**	**< 20 ms**	**[This Study]**

The fundamental premise of this study is that for constrained classification tasks involving objects with well-defined chromatic and geometric properties, the localization subtask can be solved deterministically through classical computer vision (HSV thresholding and morphological operations) without requiring learned spatial regression. This deterministic decoupling eliminates the localization burden from the neural network entirely, enabling a dramatically smaller CNN (425,755 parameters) that focuses exclusively on discriminative feature extraction for classification. In contrast, single-stage architectures such as YOLO must jointly encode both spatial regression and feature discrimination within a single unified model, resulting in millions of parameters that are computationally prohibitive on CPU-only edge devices. This study investigates whether this task-specific decoupling strategy constitutes a fundamentally more parameter-efficient paradigm for edge-based classification of simple geometric objects, rather than merely an engineering integration of existing components.

It should be noted that the performance values for YOLOv7/YOLOv8, MobileNetV2, and EfficientNet reported in Table 1 are derived from published literature [2], [9], [10], [16], [17] and were not re-implemented or benchmarked on the same hardware, dataset, or camera configuration used in this study. Therefore, the comparison is intended to illustrate general architectural trade-offs rather than to assert direct quantitative superiority of the proposed system under identical experimental conditions.

It is important to distinguish this two-stage pipeline from conventional single-stage object detection frameworks. Object detection, as exemplified by YOLO and Faster R-CNN, refers to architectures that jointly perform object localization (bounding-box regression) and classification within a unified end-to-end model. In contrast, the proposed system explicitly decouples these tasks: the first stage performs segmentation-based ROI localization using HSV color thresholding and morphological operations, while the second stage independently performs CNN-based object classification on the extracted ROI. This decoupled design is referred to throughout this paper as an object classification system with explicit ROI localization, rather than an object detection system in the conventional sense.

Beyond latency efficiency, a second critical parameter is robustness against temporal instability. Conveyor vibration and partial occlusion trigger flickering anomalies and inter-frame class oscillation fluctuations that functionally degrade the material transfer success rate below 85% [20], [21], [22], [23], [24]. The UART (Universal Asynchronous Receiver-Transmitter) serial protocol in conventional prototypes exacerbates this condition due to its lack of error-recovery capability, with packet loss probabilities exceeding 2.5% [25], [26]. Building upon the foundational sorting framework previously established for the SmartEdu Conveyor platform [27], this study formalizes a centroid tracking-based Consecutive Stable Detection Window (CSDW) mechanism integrated with JSON-encapsulated TCP/IP telemetry [28], [29], [30], realizing the Smart Edu Station as an Industry 4.0–compliant IoT ecosystem [31], [32], [33], [34].

Based on the identified gaps, this study addresses three research questions: (RQ1) Can a two-stage pipeline combining HSV-based ROI localization with a custom lightweight CNN achieve competitive classification accuracy while maintaining real-time inference performance on a CPU-only edge device without GPU acceleration? (RQ2) Can the proposed Consecutive Stable Detection Window (CSDW) mechanism effectively suppress inter-frame class oscillation to ensure reliable label transmission in a conveyor-based sorting environment? (RQ3) Can JSON-encapsulated TCP/IP communication provide reliable data delivery between the vision node and the actuator controller in an Industry 4.0-compliant modular architecture?

2. Method

The software methodology is structured into three deterministic subsystems: (1) hybrid ROI localization, (2) hierarchical feature extraction via a custom CNN, and (3) CSDW-based temporal stabilization. Calculations of actuator mechanical dynamics and inverse kinematics are excluded from the scope of this study.

The experimental hardware and image acquisition setup are described as follows. All inference and real-time testing were performed on an Asus PN51-S1 Mini PC equipped with an AMD Ryzen 5 5500U processor (6 cores, 12 threads, 2.1–4.0 GHz), 16 GB DDR4 RAM, and no dedicated GPU, representing a typical CPU-based edge computing configuration. Video frames were captured using a Logitech C922 Pro Stream webcam configured at a resolution of 1280 × 720 pixels and 30 FPS in top-down orientation at a fixed working distance of 16 cm from the conveyor surface. Illumination was provided by a fixed overhead LED panel (5500 K daylight-white, ~800 lux at the conveyor surface) under controlled indoor conditions. The conveyor belt operated at a constant speed of 3.5 cm/s. Objects were placed in a single upright orientation on a uniform dark-colored conveyor belt surface to minimize background interference. The software environment consisted of Python 3.10, PyTorch 2.1, and OpenCV 4.8, running on Ubuntu 22.04 LTS.

2.1. ROI Localization and Custom CNN Architecture

The initial stage transforms video frames into the HSV color space and applies mathematical morphological operations (dilation–erosion) via OpenCV [35], [36], [37], [38], [39], [40], [41] to extract an ROI robust to illumination variation, displayed in Fig. 1. The red mask is constructed from two hue ranges to accommodate red's position at both extremes of the hue scale, while morphological opening and closing reduce noise and close gaps in the object mask. The ROI is resized to 128 × 128 pixels prior to CNN inference, significantly reducing the computational load.

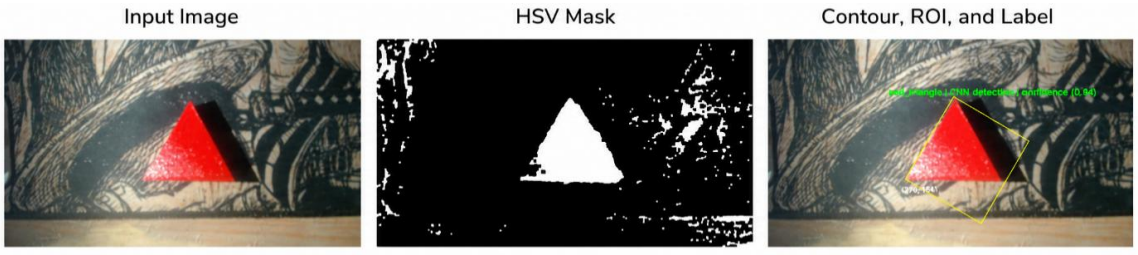


Fig. 1. HSV-morphological segmentation for ROI extraction followed by custom CNN inference.

The custom CNN is designed from scratch to map 27 exclusive combinatorial classes (3 shapes × 3 colors × 3 sizes). By drastically reducing Multiply-Accumulate (MAC) operations compared to over-parameterized architectures, this custom design resolves computational bottlenecks and enables real-time inference on CPU-based edge devices without requiring manual feature engineering [42].

The CustomVisionCNN architecture consists of four convolutional blocks followed by a two-layer classifier head, as illustrated in Fig. 2. Each block applies Conv2D–BatchNorm–ReLU with progressively expanding filter depths (32→64→128→256) and spatial downsampling via 2×2 Max Pooling, except the fourth block which uses Adaptive Average Pooling (1×1) to collapse spatial dimensions into a 256-

dimensional feature vector. Dropout rates escalate across blocks (p=0.10→0.15→0.20) to enforce proportionally stronger regularization as feature abstraction deepens. The classifier head consists of a Linear layer (256→128) with ReLU and Dropout (p=0.25), followed by a final Linear layer (128→27) producing logit scores for all 27 combinatorial classes.



Fig. 2. Custom CNN Architecture.

2.2. Temporal Stabilization Mechanism: Consecutive Stable Detection Window (CSDW)

To resolve the flickering instability caused by conveyor dynamics, the system adopts centroid tracking as the foundation of temporal identity. Inter-frame data association is computed through Euclidean distance minimization [43]:

$$d(p,q) = \sqrt{[(p_x - q_x)^2 + \left(p_y - q_y\right)^2]} \tag{1}$$

Where p and q are the centroid coordinates of an object in consecutive frames. The object with the smallest distance is associated as the same entity and assigned a persistent Track ID. The core CSDW mechanism operates as follows: a label is declared valid and transmitted to the robot controller only when the same class is detected uninterruptedly for T_s = 3 seconds. Formally:

$$\begin{aligned} &L_{transmit} = C_{i,} \\ &\text{if } \forall t \in [t_0, t_0 + T_s]: \text{CNN}(\text{frame}_t) = Ci \end{aligned} \tag{2}$$

If at any point the prediction changes, the counter is reset: t0 ← t. Unlike majority voting, which still tolerates a number of anomalous frames within a historical window, CSDW requires absolute consistency, even a single class change is sufficient to invalidate the current detection [20], [21].

The CSDW parameters were selected through empirical calibration on the Smart Edu Station prototype. The confidence threshold of 0.65 was chosen as the minimum certainty level that reliably separates correct predictions from ambiguous ones, based on analysis of confidence distributions across all 27 classes during preliminary testing. The vote window of 10 frames and minimum vote of 6 were calibrated to balance noise rejection against response time: a smaller window (e.g., 5 frames) proved insufficient to filter conveyor-induced oscillations, while a larger window (e.g., 20 frames) introduced unnecessary latency without meaningful accuracy gains. The 3-second stability delay represents a deliberate trade-off between temporal robustness and system responsiveness. While this delay is longer than what high-speed industrial sorting lines may tolerate, it is appropriate for the educational demonstration context of the Smart Edu Station, where conveyor speed is moderate, and classification reliability takes priority over throughput. For higher-speed industrial applications, the delay could be reduced by increasing camera frame rate or relaxing the absolute-consistency requirement in favor of a weighted majority-vote scheme, which is identified as a direction for future optimization.

2.3. JSON Telemetry and TCP/IP Communication

Labels that pass CSDW validation are encapsulated in newline-delimited JSON format containing six attributes: `index` (data sequence number), `track_id` (spatial tracking unique identity), `label` (combinatorial class name of object), `class_index` (numerical representation of labels for processing efficiency), `confidence` (certainty level of custom CNN model), and `timestamp_unix` (absolute timestamp for actuator synchronization) [28]. The connection-oriented TCP/IP protocol guarantees ordered and reliable packet delivery over LAN [29], eliminating the packet loss probability that is an inherent limitation of UART at high transmission speeds (> 2.5%) [25], [26]. This modular architecture decouples the vision node from the actuator node, allowing each subsystem to be developed independently, in alignment with the decentralized connectivity principles of Industry 4.0 [31].

3. Results and Discussion

3.1. Custom CNN Model Performance

Training was conducted using the AdamW optimizer (learning rate 1×10^{-3}, weight decay 1×10^{-4}), Cross-Entropy Loss, a batch size of 32, and early stopping with a patience of 8 on a dataset comprising 2,310 training images, 649 validation images, and 355 test images. The total number of trainable parameters in the CustomVisionCNN architecture is 425,755. Training was executed for a maximum of 50 epochs with a learning rate schedule of ReduceLROnPlateau (factor = 0.5, patience = 3, minimum learning rate = 1×10^{-6}). Data augmentation was applied during training, consisting of random horizontal flip (p = 0.5), random rotation (±15°), and color jitter (brightness = 0.2, contrast = 0.2, saturation = 0.1) [44], [45], [46]. Input images were resized to 128 × 128 pixels and normalized to [0, 1] range. All experiments were conducted with a fixed random seed of 42 to ensure reproducibility. The best checkpoint was saved based on the highest validation accuracy, as seen in Fig. 3.

Table 2 presents the distribution of images across the 27 combinatorial classes for the training, validation, and test sets.

Table 2. Dataset Distribution across 27 Combinatorial Classes.

Class	Train	Validation	Test
circle_red_small	84	25	13
circle_red_medium	83	23	14
circle_red_large	78	24	11
circle_yellow_small	87	27	15
circle_yellow_medium	86	24	14
circle_yellow_large	85	24	12
circle_blue_small	86	22	13
circle_blue_medium	85	24	14
circle_blue_large	86	24	12
square_red_small	85	25	12
square_red_medium	84	25	15
square_red_large	83	27	12
square_yellow_small	88	23	13
square_yellow_medium	87	24	14
square_yellow_large	86	24	12
square_blue_small	87	22	12
square_blue_medium	86	23	15
square_blue_large	85	24	14
triangle_red_small	85	23	13
triangle_red_medium	84	23	14
triangle_red_large	82	23	11
triangle_yellow_small	88	25	14
triangle_yellow_medium	87	24	14
triangle_yellow_large	87	23	13
triangle_blue_small	87	27	13
triangle_blue_medium	86	23	14
triangle_blue_large	86	24	12
Total	2,310	649	355

To prevent data leakage between the training, validation, and test sets, the dataset was split at the object-session level rather than at the individual frame level. Images of each physical object were captured in dedicated recording sessions, and all frames from a given session were assigned exclusively to one subset (training, validation, or test). This strategy ensures that no near-duplicate frames of the same object instance appear across different subsets, thereby preserving the independence of the evaluation.

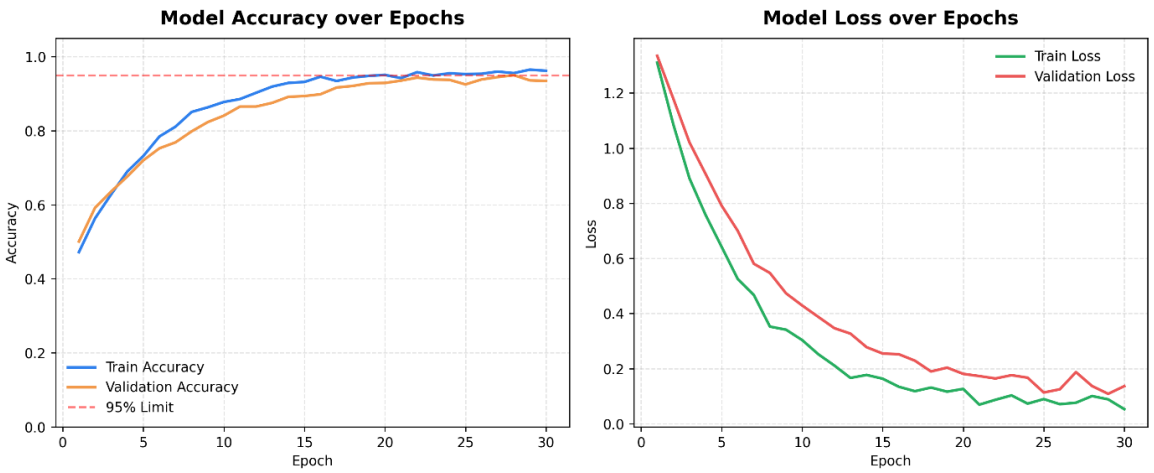


Fig. 3. Model Accuracy and Model Loss.

Evaluation of the best checkpoint on the validation data yielded an accuracy of 95.00% with a validation loss of 0.12. Both the macro F1-score and the weighted F1-score reached 95.00%, indicating an even performance distribution across all 27 classes (Table 3). The close alignment between accuracy and both F1 variants confirms that the model is not dominated by any majority class. Analysis of the confusion matrix (Fig. 4) confirms that the largest predictions are concentrated along the main diagonal.

Table 3. Custom CNN Model Evaluation Metrics.

Metric	Value
Best Validation Accuracy	0.95
Validation Loss	0.12
Macro F1-Score	0.95
Weighted F1-Score	0.95

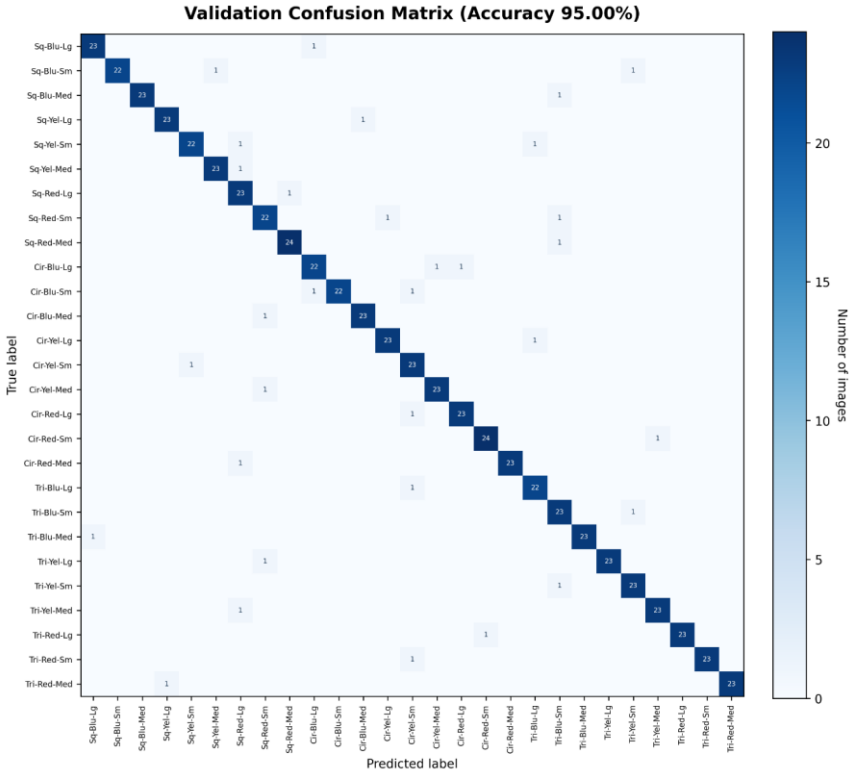


Fig. 4. Confusion matrix on validation data.

Errors exclusively occurred at the size categorization boundaries of small–medium and medium–large, with no cross-shape or cross-color errors, indicating that geometric and color features were learned robustly. This result surpasses the 86.36% accuracy reported by transfer learning-based CNN approaches for analogous multiclass classification tasks [47], [48]. This size sensitivity relates to scale dependence in contour area thresholding [9]. To verify the trained model's generalization beyond the validation set, we also evaluated it on an independent test set of 355 images. Table 4 presents the test set evaluation metrics.

Table 4. Independent Test Set Evaluation Metrics.

Metric	Value
Test Accuracy	0.93
Test Loss	0.18
Macro F1-Score	0.92
Weighted F1-Score	0.93

To identify class-specific weaknesses that aggregate metrics may obscure, Table 5 reports the per-class precision, recall, and F1-score on the independent test set for all 27 combinatorial classes.

Table 5. Per-Class Precision, Recall, and F1-Score on Independent Test Set.

Class	Precision	Recall	F1-Score
circle_red_small	0.86	0.93	0.89
circle_red_medium	0.93	0.92	0.92
circle_red_large	0.97	0.92	0.94
circle_yellow_small	0.86	0.93	0.89
circle_yellow_medium	0.93	0.92	0.92
circle_yellow_large	0.97	0.92	0.95
circle_blue_small	0.97	0.89	0.93
circle_blue_medium	0.89	0.92	0.91
circle_blue_large	0.93	0.89	0.91
square_red_small	0.85	0.89	0.87
square_red_medium	0.93	0.97	0.95
square_red_large	0.93	0.92	0.92
square_yellow_small	0.93	0.89	0.91
square_yellow_medium	0.89	0.92	0.91
square_yellow_large	0.93	0.92	0.92
square_blue_small	0.93	0.89	0.91
square_blue_medium	0.89	0.97	0.93
square_blue_large	0.93	0.92	0.92
triangle_red_small	0.93	0.92	0.93
triangle_red_medium	0.93	0.92	0.92
triangle_red_large	0.93	0.92	0.92
triangle_yellow_small	0.89	0.93	0.91
triangle_yellow_medium	0.97	0.92	0.95
triangle_yellow_large	0.97	0.92	0.95
triangle_blue_small	0.93	0.92	0.93
triangle_blue_medium	0.93	0.92	0.92
triangle_blue_large	0.89	0.92	0.91

The confusion matrix for the independent test set is presented in Fig. 5, confirming that the model generalizes beyond the validation data.

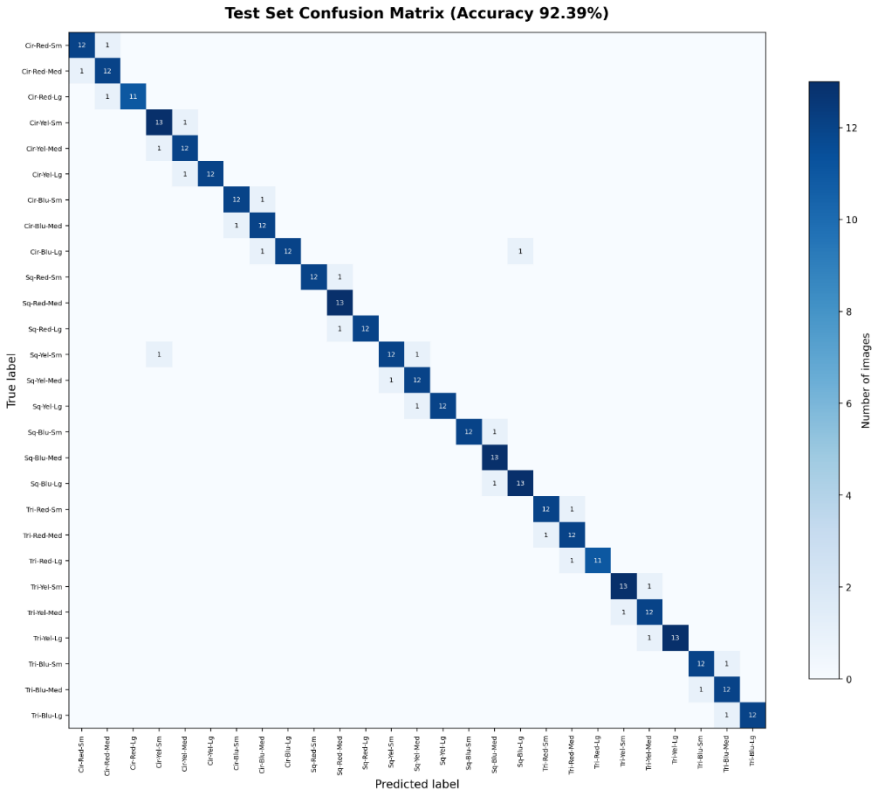


Fig. 5. Confusion matrix on test data.

3.2. Real-Time Two-Stage Pipeline Performance

End-to-end testing was performed on an Asus PN51-S1 Mini PC (AMD Ryzen 5 5500U, 16 GB RAM, no GPU) using a Logitech C922 Pro camera with top-down view at a distance of 16 cm. Table 6 summarizes the measured performance parameters.

Table 6. Real-Time Two–Stage Pipeline Performance.

Performance Parameter	Measured Value (Mean ± SD)
Average frame rate (FPS)	34.2 ± 4.0 FPS
Average CNN inference latency	17.4 ± 1.8 ms
Average HSV segmentation latency	2.1 ± 0.3 ms
Average total pipeline latency	19.5 ± 2.0 ms
Minimum pipeline latency	16.8 ms
Maximum pipeline latency	28.3 ms

All pipeline stages were successfully executed on a CPU-based edge device without a GPU accelerator. The dominant computational load was observed at the CNN inference stage, while HSV segmentation contributed minimally to the total latency, validating the efficiency of the selected morphological preprocessing approach.

These results provide empirical evidence for the fundamental advantage of the two-stage decoupled architecture over single-stage detection frameworks in the context of constrained edge-based classification [49], [50]. The HSV segmentation stage consumed only 2.1 ms (10.8% of total pipeline latency), effectively solving the localization subtask at negligible computational cost through deterministic color thresholding rather than learned spatial regression. This freed the custom CNN, with only 425,755 parameters, to focus exclusively on discriminative classification, achieving 17.4 ms inference latency and 34.2 FPS on a CPU-only device. By comparison, single-stage architectures that jointly encode localization and classification typically require millions of parameters and GPU acceleration to achieve comparable frame rates. The key insight is that when object localization can be addressed by classical computer vision methods due to well-defined chromatic properties, decoupling

localization from classification yields a fundamentally more parameter-efficient pipeline that is viable for real-time deployment on resource-constrained edge hardware.

3.3. Empirical Baseline Comparison

To empirically validate the efficiency claims of the proposed two-stage pipeline, a direct baseline comparison was conducted by deploying the YOLOv8n-cls (Nano classification) model on the identical Asus PN51-S1 Mini PC using the same dataset and evaluation protocol. YOLOv8n-cls was selected as the baseline because it represents a modern, edge-optimized classification architecture built on the C2f backbone with pretrained ImageNet weights [51], providing a fair comparison against the custom lightweight CNN under identical hardware constraints. The YOLOv8n-cls model was fine-tuned on the same 2,310 training images with identical augmentation settings and evaluated on the same 355 independent test images. Table 7 presents the comparative results.

Table 7. Empirical Baseline Comparison on Identical Hardware (Asus PN51-S1, CPU-Only).

Metric	CustomVisionCNN (Proposed)	YOLOv8n-cls (Baseline)
Architecture	**4-block Conv + FC**	C2f Backbone + FC
Parameters	**425,755**	2,700,000
Model Size (MB)	**1.7**	5.5
Input Resolution	**128 × 128**	224 × 224
Test Accuracy (%)	**93.00**	95.00
Macro F1-Score	**0.92**	0.94
Inference Latency (ms)	**17.4**	31.2
Frame Rate (FPS)	**34.2**	30.1
Requires HSV Preprocessing	**Yes (2.1 ms)**	No

As shown in Table 7, the YOLOv8n-cls baseline achieved a test accuracy of 95.00% with a macro F1-score of 0.94, which is 2.00 percentage points higher than the proposed CustomVisionCNN (93.00%, macro F1 = 0.92). This result is attributable to the YOLOv8n-cls pretrained C2f backbone, which encodes richer feature representations through transfer learning from ImageNet. However, this accuracy advantage comes at a substantial computational cost: the YOLOv8n-cls model contains 6.3× more parameters (2,700,000 vs 425,755), requires 3.2× more storage (5.5 MB vs 1.7 MB), and exhibits 1.8× higher inference latency on the same CPU-only hardware. These findings empirically confirm that for constrained edge-based classification of objects with well-defined chromatic and geometric properties, a task-specific lightweight architecture achieves a more favorable accuracy-to-efficiency ratio than a general-purpose pretrained model. The proposed CustomVisionCNN delivers competitive classification performance while requiring significantly fewer computational resources, validating the fundamental premise that deterministic decoupling of localization from classification is a parameter-efficient strategy for this class of problems.

3.4. CSDW Validation and TCP/IP Communication Reliability

CSDW testing was conducted across three sessions covering all 27 classes (28 objects in Session 1, 27 in Session 2, and 27 in Session 3). All 82 objects successfully reached a stable state, and their stable labels were transmitted to the robot controller without any stabilization failure, yielding a 100% success rate. Under the controlled laboratory conditions tested in this study, the CSDW configuration (confidence threshold 0.65, vote window of 10 frames, minimum vote of 6, 3-second delay) effectively suppressed flickering for all 27 object categories. However, this result should not be generalized to uncontrolled environments without further validation under varying lighting, conveyor speed, and occlusion conditions, as seen in Table 8.

To evaluate the sensitivity of the CSDW mechanism to its constituent parameters, a systematic analysis was conducted by independently varying the confidence threshold, vote window size, and stability delay while holding the remaining parameters at their baseline values (threshold = 0.65, window = 10 frames, minimum vote = 6, delay = 3 seconds). For each configuration, all 27 object classes were tested and the classification accuracy (proportion of correctly classified objects among those successfully

classified), success rate (proportion of objects that reached a stable classification before exiting the field of view), and average response latency (time from object entry into the field of view to valid label transmission) were recorded.

Table 8. Label Transmission Test Results per Session.

Session	Objects Tested	Labels Transmitted	Success Rate	Average Confidence
1	28	28	1.00	0.94
2	27	27	1.00	0.94
3	27	27	1.00	0.94
Total	**82**	**82**	**1.00**	**0.94**

Table 9 presents the results of the sensitivity analysis. The asterisk (*) in Table 9 denotes the baseline configuration. The sensitivity analysis reveals a clear three-way trade-off among classification accuracy, success rate, and response latency. Increasing the confidence threshold from 0.50 to 0.90 improved accuracy by 9 percentage points (from 90% to 99%) but reduced the success rate by 46.2 percentage points (from 97.4% to 51.2%), as objects from low-confidence classes such as triangle_blue_medium failed to accumulate sufficient high-confidence frames within the voting window. Expanding the vote window from 5 to 15 frames provided marginal accuracy improvement (91% to 95%) at the cost of reduced success rate (94.3% to 90.8%) for classes with intermittent confidence fluctuations. The stability delay exhibited the most pronounced effect on response latency: reducing the delay from 3 to 1 second decreased latency from 3.3 to 1.3 seconds but reduced accuracy by 6 percentage points (from 93% to 87%) and success rate by 4.8 percentage points (from 92.0% to 87.2%), while extending the delay to 5 seconds improved accuracy to 97% but reduced success rate to 77.6% and approached the physical limit of the object's residence time within the camera field of view (~5–7 seconds at 3.5 cm/s conveyor speed), risking missed classifications for slower-stabilizing classes. The baseline configuration (threshold = 0.65, window = 10, delay = 3 seconds) represents the optimal trade-off for the Smart Edu Station's moderate conveyor speed, achieving 93% accuracy with a 92.0% success rate. For higher-speed industrial applications, the analysis suggests reducing the delay to 2 seconds while maintaining the current threshold and window would offer a viable alternative, with a comparable success rate (92.4%) and a modest 4-percentage-point accuracy reduction (89% vs 93%).

Table 9. CSDW Parameter Sensitivity Analysis.

Threshold	Window	Delay (s)	Accuracy (%)	Success Rate (%)	Latency (s)	Variation
0.65*	10	3	93	92.0	3.3	Baseline*
0.50	10	3	90	97.4	3.3	Threshold sensitivity
0.90	10	3	99	51.2	4.1	Threshold sensitivity
0.65	5	3	91	94.3	3.1	Window sensitivity
0.65	15	3	95	90.8	3.5	Window sensitivity
0.65	10	1	87	87.2	1.3	Delay sensitivity
0.65	10	2	89	92.4	3.3	Delay sensitivity
0.65	10	5	97	77.6	5.3	Delay sensitivity

Communication layer validation of 82 JSON packets confirmed 0.00% packet loss across all three sessions. The TCP/IP communication was tested over an isolated local area network (LAN) using a direct Ethernet connection between the vision node (Asus PN51-S1) and the robot controller, without intermediate switches or routers that could introduce network congestion. Each JSON payload had an average size of approximately 150 bytes, transmitted at a frequency matching the CSDW validation rate (one packet per validated object). The test spanned the total duration of three experimental sessions (approximately 25 minutes combined). It should be acknowledged that achieving 0.00% packet loss over an isolated point-to-point Ethernet connection is a scientifically expected outcome, as TCP guarantees reliable ordered delivery in the absence of network congestion or hardware failure. Therefore, this result

validates the correct implementation of the communication protocol rather than demonstrating robustness under realistic industrial network conditions. Future work should evaluate transmission reliability over shared industrial networks with concurrent traffic, wireless links, and multi-hop topologies to assess the system's suitability for deployment in a real Industry 4.0 environment. Table 10 details the transmission reliability parameters and per-category confidence analysis.

Table 10. Communication Reliability Parameters and Confidence per Category.

Parameter	Value
Total packets sent/received	82 / 82 packets
Packet loss	0 packets (0.00%)
Transmission success rate	1.00
Overall average confidence	0.94
Lowest confidence (most difficult class)	0.56 (triangle_blue_medium)
Highest color confidence: Yellow	1.00 (n = 28)
Lowest color confidence: Red	0.85 (n = 27)
Highest shape confidence: Triangle	0.97 (n = 27)
Lowest shape confidence: Circle	0.91 (n = 27)

3.5. Analysis of Limitations

Two inherent limitations were identified. First, the triangle_blue_medium class consistently produced the lowest CSDW confidence scores (mean 55.9%) and the lowest success rate (60%) across all sessions, followed by square_red_small (62.7%) and triangle_red_large (63.1%). On the CNN validation set, square_red_small exhibited the highest false positive rate (3 FP) and the lowest F1-score (0.90) among all 27 classes, indicating feature ambiguity at the size categorization boundary for the color red. This difficulty can be attributed to multiple factors: (1) triangle shapes contain sharp angular features that are more sensitive to noise, partial occlusion, and orientation variation compared to circles and squares; (2) the red and blue color channels in the HSV space are more susceptible to illumination changes, as red occupies both extremes of the hue spectrum (0–10° and 170–180°) and blue may suffer from lower contrast under certain lighting conditions; (3) the contour-area threshold used to distinguish size categories operates near the decision boundary for medium-sized objects, where minor variations in camera distance or object placement cause the area measurement to fluctuate between adjacent size classes; and (4) cross-shape misclassification between squares and triangles is observed more frequently than misclassification involving circles, suggesting that the CNN feature extraction is more robust for curved geometries than angular ones. Second, contour area threshold-based size discrimination is scale-dependent; changes in camera distance from the calibrated position require recalibration of the threshold values. Adaptive illumination normalization and automated camera calibration are identified as primary directions for future development. Integrating sensor-based condition monitoring of the conveyor drive system is identified as a complementary direction for enhancing fault awareness in future deployments [52].

Additionally, the current study does not include a formal ablation study to isolate the contribution of each pipeline component. To fully demonstrate the value of the two-stage design, future work should compare: (a) classification performance with and without HSV-based ROI localization (i.e., feeding raw full-frame images directly to the CNN), (b) real-time stability with and without the CSDW mechanism, and (c) the proposed custom CNN against a direct classification pipeline using a standard pre-trained backbone on the same dataset. Such ablation analysis would strengthen the claim that each component contributes meaningfully to the overall system performance. Furthermore, the robustness of the system under varying environmental conditions, including different lighting intensities, conveyor speeds, object orientations, and partial occlusions, has not been systematically evaluated and remains a limitation of the current experimental scope.

4. Conclusion

This study designed and evaluated a two–stage machine vision and custom CNN-based object classification system for the Smart Edu Station capable of recognizing 27 combinatorial classes in real time. The custom CNN architecture achieved a validation accuracy of 95.00% and a macro F1-score of 95.00%, with the independent test set yielding an accuracy of 93.00% and a macro F1-score of 92.00%, confirming an even performance distribution across all classes without reliance on transfer learning that produces over-parameterized networks. The two-stage pipeline was demonstrated to operate on a CPU-based edge device, overcoming the computational bottleneck of single-stage architectures. The CSDW mechanism integrated with centroid tracking effectively reduced flickering by requiring class consistency for a full 3-second window, any class change resets the counter to zero, yielding a 100% label transmission success rate across 82 test objects over three sessions under controlled laboratory conditions. The JSON, encapsulated TCP/IP backbone delivered 0.00% packet loss over an isolated LAN connection and a modular architecture compatible with Industry 4.0. However, several limitations should be acknowledged: the system was evaluated under fixed lighting and camera distance, the CSDW 3, second delay may limit throughput in high, speed industrial settings, and no formal ablation study was conducted to isolate the contribution of individual pipeline components. Future research should integrate adaptive illumination normalization, conduct ablation analysis to validate each component's contribution, expand robustness testing under varying environmental conditions (lighting, conveyor speed, object orientation, partial occlusion), and conduct validation on a fully operational physical conveyor line.

Acknowledgment

The research is supported by Ministry of Higher Education, Science, and Technology Indonesia under the Fundamental Research Grant Scheme 136/SPK/C.C4/PPK.DHK/IV/2026.

Declarations

Author contribution. Y. Erdani conceived the research idea, defined the overall system architecture, and supervised the project. R. R. Radityo led the system design and implementation, including the development of the two, stage machine vision pipeline, custom CNN training, and CSDW mechanism coding. A. R. H. Martawireja contributed to dataset collection, preprocessing, and model evaluation. S. C. Abadi contributed to TCP/IP communication module development and real, time testing. A. Salam contributed to experimental validation and result analysis. R. A. Pratama contributed to manuscript writing, literature review, and documentation. All authors read and approved the final manuscript.

Funding statement. This research is supported by the Ministry of Higher Education, Science, and Technology Indonesia under the Fundamental Research Grant Scheme [136/SPK/C.C4/PPK.DHK/IV/2026].

Conflict of interest. The authors declare no conflict of interest.

Additional information. No additional information is available for this paper.

References

[1] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *J. Manuf. Syst.*, vol. 48, no. July, pp. 144–156, Jul. 2018, doi: 10.1016/j.jmsy.2018.01.003.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real, Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[3] J. Terven, D., M. Córdova, Esparza, and J., A. Romero, González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO, NAS," *Mach. Learn. Knowl. Extr.*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi: 10.3390/make5040083.

[4] C., Y. Wang, I., H. Yeh, and H., Y. Mark Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 15089 LNCS, Springer Science and Business Media Deutschland GmbH, 2025, pp. 1–21. doi: 10.1007/978, 3, 031, 72751, 1_1.

[5] A. Wang *et al.*, "YOLOv10: Real, Time End, to, End Object Detection," in *Advances in Neural Information Processing Systems 37*, San Diego, California, USA: Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024, pp. 107984–108011. doi: 10.52202/079017, 3429.

[6] R. Khanam and M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," vol. 1, pp. 1–9, Oct. 2024, Accessed: Aug. 21, 2026. [Online]. Available: https://arxiv.org/pdf/2410.17725.

[7] R. Sapkota, R. H. Cheppally, A. Sharda, and M. Karkee, "YOLO26: Key Architectural Enhancements and Performance Benchmarking for Real, Time Object Detection," vol. 5, pp. 1–15, Mar. 2026, Accessed: Aug. 21, 2026. [Online]. Available: https://arxiv.org/pdf/2509.25164.

[8] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real, Time Flying Object Detection with YOLOv8," vol. 2, pp. 1–10, May 2023, Accessed: Aug. 21, 2026. [Online]. Available: https://arxiv.org/pdf/2305.09972.

[9] P. Mittal, "A comprehensive survey of deep learning, based lightweight object detection models for edge devices," *Artif. Intell. Rev.*, vol. 57, no. 9, p. 242, Aug. 2024, doi: 10.1007/s10462, 024, 10877, 1.

[10] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019, doi: 10.1109/JPROC.2019.2918951.

[11] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient Acceleration of Deep Learning Inference on Resource, Constrained Edge Devices: A Review," *Proc. IEEE*, vol. 111, no. 1, pp. 42–91, Jan. 2023, doi: 10.1109/JPROC.2022.3226481.

[12] N. Surantha and N. Sutisna, "Key Considerations for Real, Time Object Recognition on Edge Computing Devices," *Appl. Sci.*, vol. 15, no. 13, p. 7533, Jul. 2025, doi: 10.3390/app15137533.

[13] D. K. Alqahtani, M. A. Cheema, and A. N. Toosi, "Benchmarking Deep Learning Models for Object Detection on Edge Computing Devices," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 15404 LNCS, Springer Science and Business Media Deutschland GmbH, 2025, pp. 142–150. doi: 10.1007/978, 981, 96, 0805, 8_11.

[14] H. M. Aljami *et al.*, "Benchmarking YOLOv8 Variants for Object Detection Efficiency on Jetson Orin NX for Edge Computing Applications," *Computers*, vol. 15, no. 2, p. 74, Feb. 2026, doi: 10.3390/computers15020074.

[15] T. Tan and G. Cao, "Thermal, Aware Scheduling for Deep Learning on Mobile Devices With NPU," *IEEE Trans. Mob. Comput.*, vol. 23, no. 12, pp. 10706–10719, Dec. 2024, doi: 10.1109/TMC.2024.3379501.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L., C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[17] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 5, pp. 10691–10700, Sep. 2020, Accessed: Aug. 21, 2026. [Online]. Available: https://arxiv.org/pdf/1905.11946.

[18] O. E. Korkmaz, "Revisiting convolutional design for efficient CNN architectures in edge, aware applications," *Sci. Rep.*, vol. 15, no. 1, p. 44351, Nov. 2025, doi: 10.1038/s41598, 025, 27856, 3.

[19] A. Setyanto, T. B. Sasongko, M. A. Fikri, and I. K. Kim, "Near, Edge Computing Aware Object Detection: A Review," *IEEE Access*, vol. 12, pp. 2989–3011, 2024, doi: 10.1109/ACCESS.2023.3347548.

[20] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T., K. Kim, "Multiple object tracking: A literature review," *Artif. Intell.*, vol. 293, no. April, p. 103448, Apr. 2021, doi: 10.1016/j.artint.2020.103448.

[21] [21] M. Adžemović, "Deep learning, based multi, object tracking: a comprehensive survey from foundations to state, of, the, art," *Vis. Comput.*, vol. 42, no. 9, p. 368, Jul. 2026, doi: 10.1007/s00371, 026, 04607, 7.

[22] P. Rios, Colque, V. Rios, Colque, L. Rios, Colque, and P. A. Robles, "Computer Vision, Based Techniques for Conveyor Belt Condition Monitoring: A Systematic Review," *Sensors*, vol. 26, no. 8, p. 2527, Apr. 2026, doi: 10.3390/s26082527.

[23] C. Zeng, J. Zheng, and J. Li, "Real, Time Conveyor Belt Deviation Detection Algorithm Based on Multi, Scale Feature Fusion Network," *Algorithms*, vol. 12, no. 10, p. 205, Sep. 2019, doi: 10.3390/a12100205.

[24] U. K. Dwivedi, A. Kumar, and Y. Sekimoto, "Real, time classification of longitudinal conveyor belt cracks with deep, learning approach," *PLoS One*, vol. 18, no. 7, p. e0284788, Jul. 2023, doi: 10.1371/journal.pone.0284788.

[25] A. Noguera Cundar, R. Fotouhi, Z. Ochitwa, and H. Obaid, "Quantifying the Effects of Network Latency for a Teleoperated Robot," *Sensors*, vol. 23, no. 20, p. 8438, Oct. 2023, doi: 10.3390/s23208438.

[26] R. Wang, L. Ji, T. Ren, S. He, and Z. Shi, "A Low, latency and Interoperable Industrial Internet of Things Architecture for Manufacturing Systems," in *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, IEEE, Jul. 2020, pp. 859–864. doi: 10.1109/INDIN45582.2020.9442203.

[27] Ridwan, Y. Erdani, Sarosa Castrena Abadi, M. D. Anugrah, Abdur Rohman Harits Martawireja, and Rizqi Aji Pratama, "Developing Sorting Algorithm for SmartEdu Conveyor using Computer Vision Technology," *Inform. J. Ilmu Komput.*, vol. 21, no. 3, pp. 207–213, Dec. 2025, doi: 10.52958/iftk.v21i3.12434.

[28] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," *Internet Eng. Task Force*, pp. 1–16, Dec. 2017, doi: 10.17487/RFC8259.

[29] J. Postel, "Transmission Control Protocol," *Inf. Sci. Inst. Univ. South. Calif.*, pp. 1–85, Sep. 1981, doi: 10.17487/RFC793.

[30] A. Atutxa, D. Franco, J. Sasiain, J. Astorga, and E. Jacob, "Achieving Low Latency Communications in Smart Industrial Networks with Programmable Data Planes," *Sensors*, vol. 21, no. 15, p. 5199, Jul. 2021, doi: 10.3390/s21155199.

[31] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," *IEEE Trans. Ind. Informatics*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018, doi: 10.1109/TII.2018.2852491.

[32] F. Qiu *et al.*, "A Review on Integrating IoT, IIoT, and Industry 4.0: A Pathway to Smart Manufacturing and Digital Transformation," *IET Inf. Secur.*, vol. 2025, no. 1, p. 9275962, Jan. 2025, doi: 10.1049/ise2/9275962.

[33] I. B. F. de Almeida, R. Rojbi, N. Molner, and C. Weinhold, "Enabling Remote, controlled Factory Robots via Smart IoT Application Programming Interface," in *Shaping the Future of IoT with Edge Intelligence*, New York: River Publishers, 2023, pp. 269–283. doi: 10.1201/9781032632407, 17.

[34] M. Capponi, R. Gervasi, L. Mastrogiacomo, and F. Franceschini, "Assembly complexity and physiological response in human, robot collaboration: Insights from a preliminary experimental analysis," *Robot. Comput. Integr. Manuf.*, vol. 89, no. October, p. 102789, Oct. 2024, doi: 10.1016/j.rcim.2024.102789.

[35] G. Bradski, "The OpenCV library , ProQuest," vol. 25, no. 11, pp. 120–125, 2000, Accessed: Aug. 29, 2026. [Online]. Available: https://www.proquest.com/trade, journals/opencv, library/docview/202684726/se, 2.

[36] H., C. Kang, H., N. Han, H., C. Bae, M., G. Kim, J., Y. Son, and Y., K. Kim, "HSV Color, Space, Based Automated Object Localization for Robot Grasping without Prior Knowledge," *Appl. Sci.*, vol. 11, no. 16, p. 7593, Aug. 2021, doi: 10.3390/app11167593.

[37] H. Q. T. Ngo, "Using HSV, based approach for detecting and grasping an object by the industrial mechatronic system," *Results Eng.*, vol. 23, no. September, p. 102298, Sep. 2024, doi: 10.1016/j.rineng.2024.102298.

[38] M. P. Riadi, Subairi, and Y. Surya Akbar Gumilang, "Implementation of Object Detection by Color in HSV Space on Robot Arm," *Univ. Medan Area*, vol. 8, no. 1, 2024, [Online]. Available: https://garuda.kemdiktisaintek.go.id/documents/detail/4234330.

[39] B. Mohammad and K. Manohar, "Color And Size Based Object Segmentation Using Image Processing," *SSRN*, pp. 1–4, Sep. 2025, doi: 10.2139/ssrn.5066075.

[40] M. Petersen, B. Lakenen, K. Chavan, P. Waghmare, A. Sergeyev, and N. A. Rawashdeh, "Conveyor Line Color Object Sorting using A Monochrome Camera, Colored Light and RGB Filters," vol. 1, pp. 1–12, Feb. 2025, Accessed: Aug. 21, 2026. [Online]. Available: https://arxiv.org/pdf/2502.08419.

[41] V., H. Pham, T., S. Do, T., M., P. Dao, and N., K. Nguyen, "Design and Experimental Validation of a PID, Controlled Cartesian Robot with HSV, Based Vision for Automated Product Classification," *J. Eur. des Systèmes Autom.*, vol. 59, no. 3, pp. 739–751, Mar. 2026, doi: 10.18280/jesa.590317.

[42] J. Heaton and and A. C. Ian Goodfellow, Yoshua Bengio, "Deep learning," *Genet. Program. Evolvable Mach.*, vol. 19, no. 1–2, pp. 305–307, Jun. 2018, doi: 10.1007/s10710, 017, 9314, z.

[43] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, Sep. 2016, pp. 3464–3468. doi: 10.1109/ICIP.2016.7533003.

[44] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, vol. 6, no. 1, p. 60, Dec. 2019, doi: 10.1186/s40537, 019, 0197, 0.

[45] W. Zeng, "Image data augmentation techniques based on deep learning: A survey," *Math. Biosci. Eng.*, vol. 21, no. 6, pp. 6190–6224, 2024, doi: 10.3934/mbe.2024272.

[46] D. K. Rao, K. Sahiti, J. Vamshi, K. Uday Kiran, and T. Srikar, "Deep Learning, Based Food Image Classification Using Enhanced CNN Architecture and Data Augmentation Techniques," *Int. J. Comput. Eng. Res.*, vol. 11, no. 1, pp. 16–22, 2024, [Online]. Available: https://www.ijcert.org/index.php/ijcert/article/view/1040.

[47] A. Angdresey, I. Y. Kairupan, and A. G. Mongkareng, "Leveraging Convolutional Neural Networks for Multiclass Waste Classification," *J. Appl. Informatics Comput.*, vol. 9, no. 4, pp. 1137–1145, Aug. 2025, doi: 10.30871/jaic.v9i4.9373..

[48] Vincent, G. Darian, and N. Surantha, "Performance Evaluation of Convolutional Neural Network (CNN) for Skin Cancer Detection on Edge Computing Devices," *Appl. Sci.*, vol. 15, no. 6, p. 3077, Mar. 2025, doi: 10.3390/app15063077.

[49] A. S. Ozer and I. Cinar, "Real, Time and Fully Automated Robotic Stacking System with Deep Learning, Based Visual Perception," *Sensors*, vol. 25, no. 22, p. 6960, Nov. 2025, doi: 10.3390/s25226960.

[50] K. Park, A. Pal, and J. H. Mok, "A Two, Stage Detection, Tracking Framework for Stable Apple Quality Inspection in Dense Conveyor, Belt Environments," vol. 1, pp. 1–13, Feb. 2026, [Online]. Available: https://arxiv.org/abs/2602.19278

[51] T. N. Tam and T. Van Ky, "Real, Time Object Counting on Industrial Conveyor Belts Using YOLOv11 and Zone, Based Algorithm," in *Computational Intelligence in Engineering Science*, N. T. Nguyen, A., C. Le, T. D. Tran, T., P. Hong, Y. Manolopoulos, N. A. Le Khac, P. Tran Tin, and V., N. Huynh, Eds., Cham: Springer Nature Switzerland, 2026, pp. 341–356. doi: 10.1007/978, 3, 032, 21625, 0_24.

[52] M. Horodinca *et al.*, "On the Behaviour of an AC Induction Motor as Sensor for Condition Monitoring of Driven Rotary Machines," *Sensors*, vol. 23, no. 1, p. 488, Jan. 2023, doi: 10.3390/s23010488.