Real-time multi-object recognition and classification on resource-constrained devices for automated robots

Phi Van Lam[1*], Vu Duc Du[1] and Tran Thi Lan[1]

[1] *University of Transport and Communications*
[*] *Corresponding author E-mail:pvlam@utc.edu.vn*
DOI: https://doi.org/10.64032/mca.v30i2.420

Abstract

In order to solve the critical challenges of deploying classical CNNs on resource-constrained edge devices—such as excessive computational demands, high latency, and reduced accuracy in complex environments—this paper proposes a robust, decoupled real-time multi-object recognition and classification framework for automated robots. Focused on the dynamic conditions of the Robocon 2026 competition, which demands distinguishing 31 distinct object categories, we introduce an optimized two-stage computer vision architecture to overcome the parameter bloat and accuracy degradation typical of single-stage multi-class detectors. The proposed pipeline leverages the optimal anchor-free design of the lightweight YOLOv8n model exclusively for rapid ROI extraction and introduces a custom Separable CNN equipped with a Flatten and Dense(256) classification head, compressed to only 2.11 million parameters for precise morphological feature identification. To enhance model resilience against geometric distortions and minimize the input payload without sacrificing crucial discriminative cues, morphological preprocessing techniques including Gray Padding and grayscale spatial transformation are implemented. Furthermore, to effectively balance the trade-off between detection speed and accuracy, the system is strictly accelerated through Int8 network quantization and Intel's OpenVINO toolkit, significantly reducing hardware latency. Real-world processing speeds, hardware execution latency, and classification accuracy are utilized as primary benchmarks for evaluation. Experimental results conducted on a low-power Intel Surface Go 2 platform demonstrate that the proposed architecture achieves an impressive 96.15% accuracy while maintaining a stable real-time end-to-end pipeline processing speed of 30-35 FPS. A comprehensive comparison against state-of-the-art lightweight architectures, including MobileNetV3 Small, SqueezeNet 1.1, and EfficientNet Lite0, shows that the proposed model outperforms existing solutions by delivering higher detection precision with drastically lower computational overhead. These outcomes confirm that our methodology provides a highly feasible, scalable, and efficient solution well-suited for deployment in intelligent edge robotics.

Keywords: *YOLOv8n; Separable CNN; OpenVINO; Albumentations; Robocon 2026.*

Abbreviations

MB	Megabytes
mAP50	Mean Average Precision at 0.5
FPS	Frames per second
AI	Artificial Intelligence
CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
R-FCN	Region-based Fully Convolutional Network
ResNet	Residual Neural Network
SSMD	Single Shot MultiBox Detector
YOLO	You Only Look Once
ROI	Region of Interest
iGPU	Integrated Graphics Processing Unit
KFS	Knowledge Fragment Stone
ABU	Asia-Pacific Broadcasting Union

1. Introduction

In recent years, the explosion of artificial intelligence (AI) and computer vision has generated remarkable advancements in the field of mobile robot automation. The ABU Asia-Pacific Robot Contest (Robocon) 2026 introduces a compelling theme centered around the recognition and collection of ancient Oracle Bone Script characters. According to the official competition rulebook [1], automated robots are tasked with navigating a dynamic arena to identify, retrieve, and correctly sort specific physical containers bearing these intricate symbols. At the core of this challenge lies the technical requirement to construct a vision system capable of accurately recognizing and classifying 31 distinct categories of objects (Figure 2)—comprising genuine object groups (*real_01* to *real_15*), deceptive fake object groups (*fake_01* to *fake_15*), and a special team label (*R1*)—within a competitive environment characterized by continuously fluctuating lighting conditions and varying angles.

(a) (b)

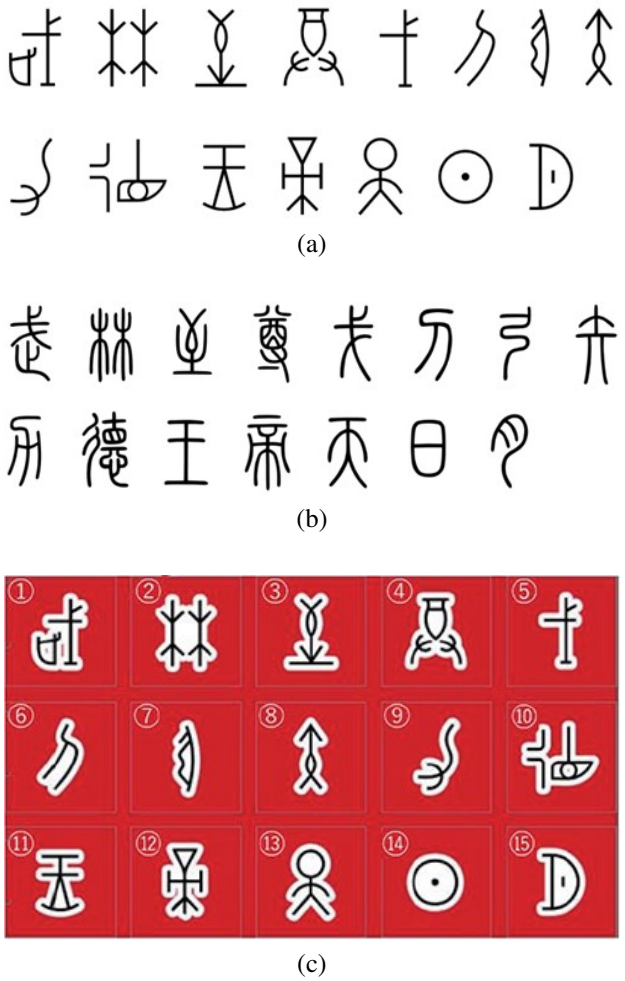
Figure 1: Symbols affixed to the R1 KFS: (a) Red box, (b) Blue box.

Detailed in the technical specifications, these objects are standard cubic boxes as shown in Figure 1 with outer dimen-

Received on DD-MM-YYYY. Accepted on DD-MM-YYY.

sions of 350 × 350 × 350 mm (inner dimensions: 344 × 344 × 339 mm). Such consistent geometric sizing allows the robot's vision system, operating through the Intel Surface Go 2 camera, to effectively estimate spatial scale and accurately localize target features at varying distances.

Although modern CNN models such as ResNet or YOLO possess superior feature extraction capabilities to evaluate complex inputs, they are naturally accompanied by network structures containing tens of millions of parameters. For instance, the robust performance of the YOLOv8 architecture has recently been demonstrated as highly effective even in challenging contexts, such as detecting small objects within remote sensing imagery under complex backgrounds [2]. However, this massive computational intensity leads to physical computational bottlenecks when deployed on highly mobile edge devices that are strictly constrained in terms of energy and memory (such as the Surface Go 2 using an Intel Pentium 4425Y CPU).



(a)

(b)

(c)

Figure 2: Visual patterns distinguishing the object classes: (a) Real KFS featuring valid Oracle Bone characters, (b) Fake KFS featuring visually similar but meaningless random stroke patterns, and (c) 15 distinct numbered red templates corresponding to the predefined Real set.

Deploying machine learning directly on such edge infrastructures is increasingly critical for real-world applications. In the context of smart urban management, for instance, deep learning models integrated with open-source frameworks like OpenCV and TensorFlow Lite have facilitated real-time license plate extraction and recognition efficiently at parking lot gates [3].

To overcome the resource barriers inherent to embedded environments, compact network architectures like SqueezeNet or MobileNet have been continuously reformed. Recent implementations of improved MobileNetV3 networks have shown exceptional industrial capability in coal and gangue recognition tasks, achieving high precision while drastically reducing network layers and computational load [4]. Yet, regarding specific problems like processing customized multi-channel or single-grayscale formats on edge devices, generic structures often retain architectural redundancies.

Despite these redundancies, optimizing models for real-time edge processing remains a priority under strict dynamic conditions. In transportation safety, optimized CNN architectures processing dashcam data execute traffic light classification with minimal latency, instantly rendering visual alerts on LED displays for trailing vehicles [5].

Beyond architectural adjustments, deploying dedicated software tools for hardware acceleration is a critical factor to meet these stringent speed constraints. Research on optimizing trained neural networks (e.g., SSD_MobileNet_V2_COCO) leveraging the OpenVINO Toolkit has demonstrated an average performance increase of 130 times, confirming that such accelerators are highly appropriate and stable for Intel processors [6].

When combining software accelerators with lightweight architectures, edge-level reliability naturally extends into stringent industrial quality control. Dedicated computer vision models have been successfully utilized to classify extremely fine defects on electronic component suction nozzles in electric vehicle manufacturing lines, realizing near-perfect accuracy and solidifying robotic automation [7].

Furthermore, within human-centric kinematics, embedded computer vision demonstrates substantial potential. For example, autonomous image processing algorithms applied to physical two-wheeled robotic canes actively support visually impaired users in maintaining balance via precise motion trajectory estimations [8].

In recent studies, the deployment of lightweight object detection models on edge devices has gained significant attention across diverse applications. For instance, foundational guidelines for balancing the speed and accuracy trade-offs in modern convolutional detectors like Faster R-CNN, R-FCN, and SSD were established in [9]. Building on these principles, modern YOLO architectures have been highly optimized for edge and resource-constrained environments. The feasibility of YOLOv7-tiny on embedded platforms was demonstrated in [10], achieving up to 30 FPS on the Jetson Xavier NX. Similarly, YOLOv3 and SSMD were deployed via TensorFlow Lite on a Raspberry Pi for real-time dog detection in [11], achieving 94% accuracy with YOLOv3. Furthermore, FPGA-based solutions, such as the system proposed in [12] using YOLOv3 Tiny on the Kria KV260, have shown remarkable efficiency in Advanced Driving Assistance Systems (ADAS), delivering 15 FPS while consuming only 3.5W of power. Beyond standard environments, specialized YOLO derivatives have been developed for challenging conditions. A lightweight model, YOLOv8-AES, was introduced in [13], achieving continuous 500 FPS processing for underwater object detection, while the work in [14] proposed GMS-YOLO (based on YOLOv10) for

pedestrian and vehicle detection under dense foggy weather, securing stable high-FPS inference on Jetson Nano. Highlighting the efficacy of hybrid architectures, a two-stage YOLO-CNN pipeline (combining YOLOv11n and EfficientNet-B0) was designed in [15] to efficiently localize and classify lithium-ion battery defects. These advancements underscore the critical relevance of developing task-specific, lightweight hybrid architectures for real-time edge processing.

Inspired by these diverse and scalable integrations of embedded computer vision—from industrial inspection to dynamic robotic movement—this research proposes a functional division processing network structure (**Two-stage pipeline**):

While object recognition and classification are conventionally solved by single-stage models, deploying a unified YOLO architecture to simultaneously detect and classify all 31 visually similar and complex Oracle Bone Script classes poses severe challenges on resource-constrained devices. A single-stage YOLO model tasked with resolving 31 fine-grained classes requires a significantly deeper network to distinguish intricate geometric strokes, inherently leading to bloated parameter sizes and unacceptable inference latency on low-power CPUs/iGPUs like the Intel Pentium 4425Y. Furthermore, processing highly distorted classes within a unified pass often degrades localization performance due to the coupled nature of detection and classification heads. To resolve this, we propose a two-stage architecture that strictly separates coarse localization from fine-grained morphological classification.

In our primary detection stage, YOLOv8n was selected as the ROI extractor based on three converging criteria established in the published literature. First, YOLOv8n's anchor-free design and C2f backbone have been demonstrated to achieve superior localization precision over YOLOv5n and YOLOv7n on small and medium objects while maintaining a significantly reduced parameter footprint [16]. Second, unlike YOLOv8s — which delivers marginal mAP gains at the cost of nearly 3.5× more parameters and substantially higher inference latency — YOLOv8n represents the optimal operating point on the accuracy-efficiency Pareto frontier for iGPU-constrained deployments [10]. Third, YOLOv8n is the earliest YOLO generation with native end-to-end OpenVINO INT8 export support via Ultralytics, enabling direct iGPU acceleration on the Intel UHD 615 without custom kernel modifications — a compatibility requirement that eliminates YOLOv5n and YOLOv7n from consideration for this deployment target [6]. Instead of forcing YOLOv8n to perform the heavy lifting of classification, it acts solely as a rapid background filter to extract Regions of Interest (ROI). The subsequent fine-grained classification is strictly offloaded to our highly compact custom **Separable CNN** (∼2.11M parameters). Simultaneously, employing software-level streamlining processes (Neural Network quantization to the Int8 standard and leveraging the OpenVINO accelerator engine), the architectural system excellently achieves a convergence point with real-time resolution quality without sacrificing parameter stability on low-end iGPU hardware.

In summary, the novelty and main contributions of this paper are as follows:

- **Novel Two-Stage Edge Framework:** Novel Hardware-Aware Two-Stage Decoupling for Fine-Grained Ancient Script Classification: To our knowledge, this is the first work to deploy a decoupled YOLO+CNN pipeline specifically optimized for 31-class Oracle Bone Script recognition on sub-watt iGPU hardware. Unlike prior two-stage works [15] targeting industrial static environments, this system operates under dynamic competition conditions with continuous geometric distortion and illumination variance.
- **Grayscale-Optimized Separable CNN with Empirically Validated Color Independence:** We demonstrate — through controlled ablation (Table 2) — that for stroke-topology classification tasks, 1-channel grayscale input surpasses RGB (96.15% vs 95.80%) while reducing parameters by 67% and tripling inference speed. This challenges the common assumption that color channels are always beneficial for visual classification.
- **Skip-Frame + Kalman Synergy for iGPU Bottleneck Recovery:** We propose and validate an asymmetric scheduling strategy that exploits the 97× speed differential between the YOLO bottleneck (18.7 FPS) and the CNN classifier (372 FPS standalone), effectively doubling system throughput from 18.7 FPS to 30–35 FPS without retraining or hardware upgrade — a generalizable strategy for any two-stage pipeline on bandwidth-constrained iGPU devices.

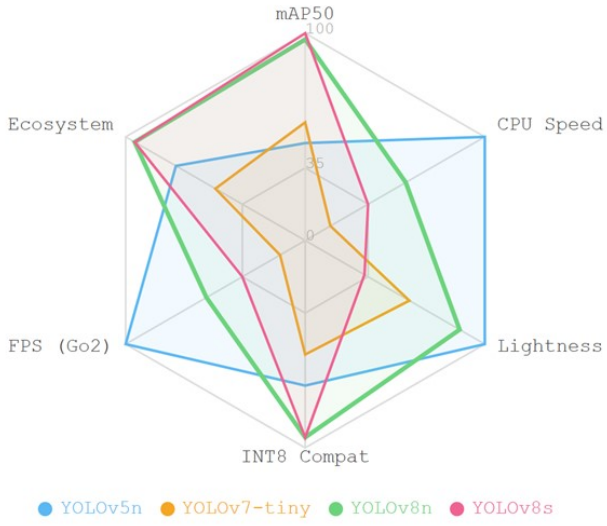


Figure 3: Optimized stratified data processing flowchart.

As illustrated in Figure 3, YOLOv8n achieves the most balanced profile across all five evaluation criteria. While YOLOv8s matches YOLOv8n in mAP50, INT8 Compat, and Ecosystem support, it scores substantially lower on Lightness and FPS (Go2) due to its 3.5× larger parameter count — rendering it unsuitable for sustained real-time inference on the thermally constrained Surface Go 2 iGPU. YOLOv5n leads on Lightness and CPU Speed but suffers on mAP50 and lacks native OpenVINO INT8 export support, introducing additional integration overhead. YOLOv7-tiny presents the weakest overall profile, particularly on INT8 Compat and FPS (Go2), confirming its architectural incompatibility with the target deployment environment. YOLOv8n uniquely satisfies all five criteria simultaneously — delivering competitive mAP50 (98.71%), the highest FPS on Surface Go 2 among accurate models (18.7 FPS pre-Skip-Frame), native INT8/OpenVINO compatibility, and a compact 3.2M parameter footprint — establishing it as

the optimal detection backbone for the proposed Two-stage pipeline

2. Proposed methodology

2.1. Two-stage recognition architecture

The system is designed with a parallel pipeline mechanism, leveraging multi-threading processing capabilities to optimize performance on the Surface Go 2: **Detection Stage:** The YOLOv8n model acts as a "confidence filter". Instead of performing complex classification computations on the full 1920×1080 image frame, YOLO only scans the frame to search for suspected regions containing objects. This serves as a coarse screening step that eliminates over 90% of the uninformative background image areas, thereby freeing up resources for the subsequent stage. **Classification Stage:** After being extracted and standardized according to the procedure in Section 2.2, the ROI is fed into a custom deep architecture Separable CNN. Thoroughly separating the tasks of "confidence region detection" and "entity identification" allows the post-check model to focus all computational resources on extracting in-depth morphological features within a narrow ROI space. By utilizing Depthwise Separable Convolution blocks paired with a Flatten and Dense(256) classification head, the v2.3 model with a scale of $\sim$2.11M parameters achieves feature representation capabilities superior to those of traditional flat networks. Owing to the combination of optimal network depth and the Label Smoothing technique, the system can flawlessly distinguish between 31 item categories, achieving an empirical accuracy of up to 96.15% even under conditions where the target is tilted or subjected to complex geometric distortions.

2.2. Data collection and polarization preprocessing

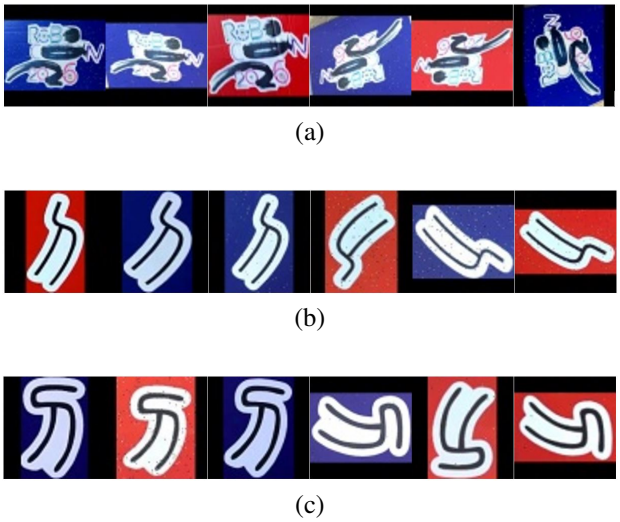
To accurately distinguish between genuine and decoy objects, the vision system must compute and differentiate features from their respective markers.

As illustrated in Figure 2, the genuine objects (Real KFS) are labeled with valid ancient Oracle Bone Scripts, while the decoys (Fake KFS) exhibit random, meaningless stroke patterns explicitly designed to confuse the model. This subtle geometric and structural variance makes naive end-to-end classification highly prone to errors. Therefore, instead of requiring an overly generalized end-to-end structure, the synthesized dataset is divided into two focused structures:

Spatial recognition dataset (Stage 1 - ROI): Contains static parameters describing the bounding box enclosing the object. The background task of YOLOv8n is to eliminate the background rather than segment the object, completely removing 90% of valueless empty space.

Differential classification dataset (Stage 2 - CNN): Establishes a repository comprising 27,900 ROI image samples. The main objective is to train a differential analysis architecture on the similar character connecting segments between Real, Fake, and R1. To maintain standard geometric spatial structure, data normalization algorithms are used:

- **Square Padding:** Padding 0-pixel bands around the short edge boundaries of the image to reconstruct an absolute square image grid instead of linear size compression (Resize), eliminating mechanical distortion noise.
- **Grayscale Spatial Transformation:** Converting from RGB (3-channel) to single-channel luminance representation eliminates 66% of the convolution operations at the CNN input stage. Critically, this conversion incurs no discriminative loss for the present classification task: the 31 target classes are defined exclusively by the morphological topology of Oracle Bone Script stroke patterns — geometric attributes entirely encoded in luminance gradients rather than chromatic channels. Although the physical containers exhibit distinct red and blue coloration (Figure 1), this color differentiation pertains solely to the R1 team-label identification at the robot hardware level and is not a feature the CNN is tasked to resolve. The 31-class boundary is drawn entirely along stroke geometry, for which grayscale luminance provides equivalent — and in practice superior — discriminative signal by concentrating the network's capacity on edge density rather than irrelevant hue variance caused by arena lighting fluctuations.
- **Interpolation Data Augmentation:** Deploying Albumentations (Rotations $\pm 45^\circ$, Perspective Transform) to create distributional invariance.



(a) (b) (c)

Figure 4: Illustration of character bounding box allocation extracted across different classes: (a) Image region extracted from ROI for R1 class, (b) Image region extracted from ROI for real_06 class, and (c) Image region extracted from ROI for fake_06 class.

Dataset Collection Protocol. The differential classification dataset was assembled under representative real-world conditions reflecting the operational environment of the Robocon 2026 competition arena. Image capture was conducted across three distinct lighting scenarios: standard indoor fluorescent lighting ($\sim$300–500 lux), simulated competition-arena spotlighting with directional glare ($\sim$800–1,200 lux), and low-ambient shadow conditions ($\sim$50–100 lux). Samples were acquired from five discrete camera viewpoints relative to the object face: frontal (0°), left/right lateral tilt ($\pm 20^\circ$), top-down oblique ($\sim 45^\circ$), and extreme skew ($\sim 60^\circ$–70°), reflecting the angular variance encountered during arena traversal. All images were captured using the integrated Intel Surface Go 2 webcam at a working distance of 30–80 cm, consistent with the robot's operational engagement range. The finalized dataset comprises 27,900 ROI image samples distributed uniformly

across 31 classes, yielding approximately 900 samples per class. The dataset was partitioned using a stratified three-way split to preserve class balance across all subsets: 70% for training (∼19,530 samples), 15% for validation (∼4,185 samples), and 15% for held-out testing (∼4,185 samples). The training subset was subjected to online data augmentation via an Albumentations pipeline incorporating: SafeRotate ($\pm 45^\circ$, $p = 0.8$), HorizontalFlip ($p = 0.5$), VerticalFlip ($p = 0.5$), Perspective Transform (scale 0.05–0.12, $p = 0.7$), RandomBrightnessContrast ($p = 0.5$), GaussianBlur ($p = 0.3$), and GaussNoise ($p = 0.3$), forging distributional invariance against the geometric distortions and luminance fluctuations characteristic of the competition environment. The validation subset received no augmentation and was used exclusively for early stopping (patience = 20) and learning rate scheduling (ReduceLROnPlateau, factor = 0.5, patience = 7). The test set was withheld entirely throughout all training and hyperparameter tuning stages and served solely as the basis for final performance reporting, including the 96.15% classification accuracy and the confusion matrix analysis presented in Section 3.1.

2.3. Static region proposal stage (YOLOv8n - object Detection)

YOLO represents a prominent example of a one-stage detector architecture, performing bounding box coordinate and class probability predictions through a single sweep of the neural network.

- **Backbone and Neck Mechanism:** The YOLOv8n version utilizes the C2f (Cross-stage Partial Bottleneck with two convolutions) structure in its Backbone to extract multi-level features. This mechanism enables the network to capture both detailed features (small objects) and contextual features (field location).
- **Multi-component Loss Function:** YOLO optimizes the training process by synthesizing multiple loss functions: L_{box} (bounding box precision using IoU), L_{cls} (classification label loss), and L_{dfl} (distribution focal loss). Achieving an mAP50 score of up to 98.71% verifies that these loss functions have optimally converged on the Stage 1 dataset.

2.4. Optimized separable CNN design for distorted character recognition

After YOLOv8n performs ROI extraction, the study proposes a Depthwise Separable CNN architecture to replace traditional flat architectures. This architecture is designed based on three core principles to balance feature depth and performance on the embedded Surface Go 2 device:

a) Optimizing depth and parameter density with Separable Convolution

Instead of using standard convolution layers that consume significant resources, the model utilizes a deep hierarchical architecture with 3 Separable Blocks followed by a Dense(256) classification head. This choice is based on a technical analysis of practical complexity:

- **Visual Complexity:** The 31 character classes in Robocon 2026 are not merely static shapes but include variations rotated by 45°, skewed, and heavily noised by the Albumentations augmentation suite. Therefore, the network requires greater depth to extract robust Morphological features.
- **Superior Parameter Density:** With a scale of ∼2.11M parameters, the model possesses a feature representation capability 15 times stronger than simple flat networks, allowing it to learn the "differentials" between characters with similar strokes from various perspectives.
- **Detailed Structure:**
  - *Block 1 & 2:* Utilizes 32 and 64 SeparableConv2D (3×3) filters. Role: Extracts primary features such as edges, corners, and initial stroke structures.
  - *Block 3 & 4:* Upgrades to 128 filters to capture high-level features and spatial invariance. The output is flattened into an 8,192-dimensional vector, then fed into a Dense(256) layer with L2 regularization ($\lambda = 0.002$)—this is the main source (∼2.10M) of the total 2.11M parameter budget.
  - *Normalization:* Each block integrates Batch Normalization and an ELU (Exponential Linear Unit) activation function, which stabilizes the Gradient, prevents dying neurons, and accelerates convergence speed.
  - *Regularization:* Integrates Dropout (0.5) and Label Smoothing (0.1) at the final layer to soften the decision boundaries, granting the model excellent anti-overfitting capabilities against noisy data.

b) Dense(256) Classification Head and Parameter Control Mechanism

This is the pivotal point determining the model's parameter budget. After 3 MaxPooling operations, the feature map shrinks to $8 \times 8 \times 128$, creating an 8,192-dimensional vector after the Flatten layer. The Dense(256) layer fully connects this vector, generating ∼2.10M parameters ($8,192 \times 256$ + bias)—accounting for over 99% of the total parameters. L2 regularization ($\lambda = 0.002$) is directly integrated into this layer to control overfitting without adding extra layers:

- **Technical Benefits:** The Dense(256) design coupled with Dropout(0.5) trailing behind establishes a controlled bottleneck: it forces the network to compress all morphological information of the 31 character classes into a 256-dimensional space, thereby assembling a compact, highly discriminative feature representation prior to the Softmax output.
- **Training Configuration:** The model is optimized using Adam (learning rate = 0.0005) with a 20% validation split (train: ∼22,320 samples, validation: ∼5,580 samples) for a maximum of 150 epochs. Early Stopping (patience = 20) and ReduceLROnPlateau (factor = 0.5, patience = 7) ensure stable convergence. The 96.15% accuracy is evaluated on the held-out test set (∼4,185 samples), which was withheld entirely from the training and hyperparameter tuning process.

c) Grayscale Preprocessing and 1-channel Input Optimization

Converting from the RGB color space (3 channels) to Grayscale (1 channel) at an input size of 64×64 pixels yields direct benefits for resource-constrained devices:

- **Computational Load Reduction:** Eliminates 66% of the convolution operations at the Input layer, entirely focusing

resources on learning stroke morphology.

- **Stability:** The system becomes "immune" to hue variations and color noise caused by excessive lighting on the competition arena. To empirically validate this design choice, a controlled ablation was conducted comparing the RGB-input baseline (3-channel, FP32, standard Flatten architecture) against the proposed 1-channel Grayscale Separable CNN under identical training conditions. As reported in Table 2, the Grayscale configuration achieves 96.15% accuracy — surpassing the RGB baseline (95.80%) by 0.35 percentage points — while simultaneously reducing parameter count from ∼6.5M to ∼2.11M and increasing pipeline speed from 5.2 FPS to 18.5 FPS. This confirms that RGB color channels not only fail to contribute additional discriminative information for stroke-based Oracle Bone classification, but actively introduce noise through arena lighting color casts, which the grayscale transformation inherently suppresses. The robustness of this design against lighting variance is further corroborated by the dataset collection protocol, which explicitly incorporated three distinct illumination conditions spanning a 50–1,200 lux range during data acquisition (Section 2.2). The marginal accuracy gap of 0.35 percentage points between the RGB baseline and the Grayscale model directly quantifies the noise contribution of chromatic lighting variance — confirming that grayscale conversion effectively decouples classification performance from arena illumination fluctuations.

Comparison with Standard Architectures:

- **MobileNetV2 (3.4M params):** The architecture is excessively wide and has numerous Residual connections, causing cache latency on the weak Intel Pentium CPU.
- **SqueezeNet (1.2M params):** Uses the Fire module to compress parameters but lacks the necessary depth to learn perspectively distorted characters.
- **Proposed Model (∼2.11M params):** Optimally designed following a Sequential pipeline, fully leveraging the vector computing power of the Intel iGPU via OpenVINO, achieving 30-35 FPS with a real-world accuracy of 96.15%.

Aggregated Technical Specifications:

- Total parameters: ∼2,111,054 (2.11M).
- Central activation function: ELU.
- Model footprint (Int8 format): ∼2.9 MB.

The concrete layout configuration and functional role for each layer in this Sequential design are fully described in Table **??**.

2.5. Edge inference hardware implementation optimization

Without the optimizations described in this section, direct deployment of the full YOLOv8n pipeline on the Surface Go 2 yields only 5.2 FPS (Table 2, Baseline) — caused by FP32 convolution overhead on the Pentium 4425Y and iGPU memory bandwidth saturation — falling far below the 30 FPS real-time threshold. The three-tier optimization strategy proposed here (INT8 quantization, OpenVINO layer fusion, and Skip-Frame scheduling) collectively recovers stable 30–35 FPS without accuracy loss.

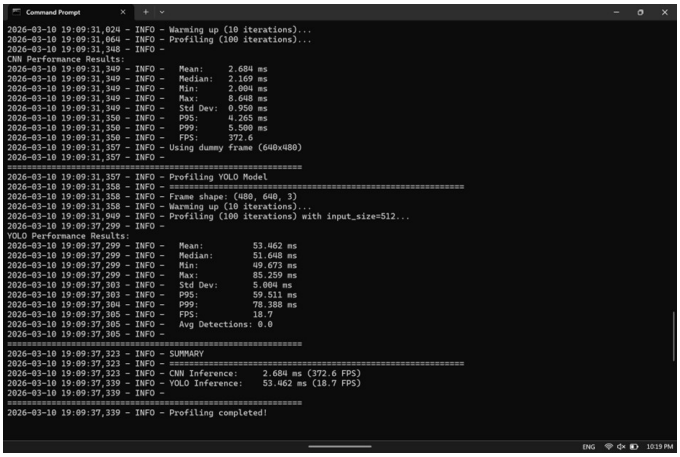


Figure 5: Optimized stratified data processing flowchart.

Pipeline FPS (Figure 5)reflects the complete end-to-end system incorporating Skip-Frame scheduling (SKIP_FREQ=2), where YOLOv8n operates at 18.7 FPS and the CNN classifier, consuming only 2–3ms per inference, runs continuously at up to 372 FPS standalone. The Kalman Filter compensates positional estimates during YOLO idle frames. Skip-Frame scheduling is a necessary architectural decision arising from the inherent asymmetry between the two stages: YOLOv8n is the sole bottleneck on the Surface Go 2 iGPU, capped at 18.7 FPS by memory bandwidth constraints, while the CNN (372 FPS standalone) would otherwise idle 97% of the time awaiting new ROI coordinates. By decoupling the stages—YOLO updates bounding boxes every 2 frames while the CNN classifies the latest available ROI continuously—and applying a Kalman constant-velocity prediction during YOLO-idle frames to maintain spatial validity, the three-component synergy effectively doubles the raw YOLO ceiling to a measured steady-state throughput of 30–35 FPS, satisfying the real-time threshold required for smooth PID-based robot trajectory control.

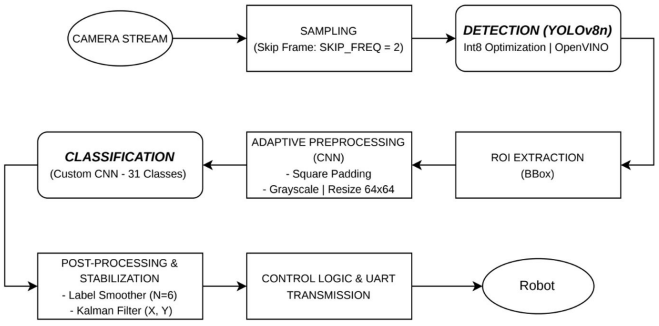


Figure 6: Optimized stratified data processing flowchart.

Int8 Quantization: Compressing the 32-bit floating-point (FP32) convolution matrices down to Int8 integer variables (-128, 127). This shrinks the CNN network weight to only ∼2.9MB (more than halved) and triples the AVX2 instruction set throughput.

OpenVINO Engine: Integrating Layer Fusion to quench implicit latency between sequential functions (Convolution & ELU) into a single duplicated Kernel, eliminating local RAM read/write overhead.

Table 2: Ablation study results isolating the performance of optimization components.

Model Configuration	Parameters	Pipeline (FPS)	Accuracy (%)	Notes
Baseline (RGB, Flatten, FP32)	∼6.5 M	5.2	95.80	Unoptimized
Grayscale + Separable Conv	∼2.11 M	18.5	96.15	3× channel reduction
Int8 Quantization (via OpenVINO)	∼2.11 M	30.5	96.15	iGPU acceleration

* All configurations evaluated on identical hardware: Surface Go 2, Intel Pentium Gold 4425Y, Intel UHD Graphics 615.

Skip Framework & Tracking Filter: YOLO scans the qualitative region at an interleaved frequency of a 1/2 cycle (15 FPS), while the 1x Grayscale CNN takes an ultra-micro load (5-6 ms) for continuous checking. The Hard Box lock containing the smallest horizontal axis coordinate (Leftmost target lock) runs comprehensively with the Smoothing mechanism of a 6-Frame Filter array to suppress rapid control Jitter Labels via a PID Kalman Filter digital filtering algorithm.

As illustrated in Figure 6, the optimized stratified data processing pipeline commences with capturing the camera stream, followed by a sampling phase that reduces computational overhead by skipping frames (SKIP_FREQ=2). The YOLOv8n object detection model, accelerated by Int8 quantization and OpenVINO, then extracts the regions of interest (ROI). These extracted bounding boxes undergo adaptive pre-processing—specifically square padding, grayscale conversion, and resizing to a standardized 64×64 dimension. Subsequently, the custom CNN model executes a precise character classification to determine the specific class among 31 categories. To guarantee output robustness in dynamic scenarios, the post-processing stage incorporates a label smoother operating over a 6-frame window and a Kalman filter to stabilize the tracking coordinates (X, Y) before the processed target logic is relayed via UART to the robot's hardware.

3. Experimental results and discussion

3.1. Model convergence and classification performance

(a) (b) (c)

Figure 7: Real-world testing under challenging perspectives: (a) real_12, (b) real_03, (c) real_10.

Figure 7 presents the direct testing results on actual physical object models, categorized into three distinct cases: (a) correctly classifying *real_12* despite partial glare interference, (b) accurately identifying *real_03* with high confidence at an oblique angle, and (c) isolating *real_10* from background clutter. In all scenarios, the user interface systematically displays a bounding box encompassing each object, coupled with its classification label (*real_xx* or *fake_xx*) and the corresponding confidence score. The outcomes demonstrate that the system consistently distinguishes between authentic Oracle Bone characters (real) and random decoys (fake) even under variable tilt angles and fluctuating luminance conditions. Concurrently, the integrated Label Smoother filter ensures the displayed labels do not flicker across consecutive frames.

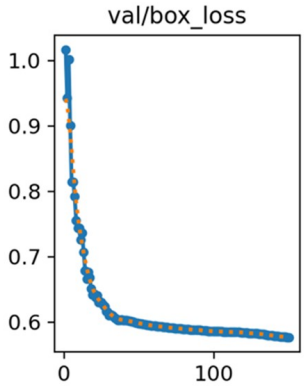


Figure 8: Validation box loss curve of the YOLOv8n detector during training.

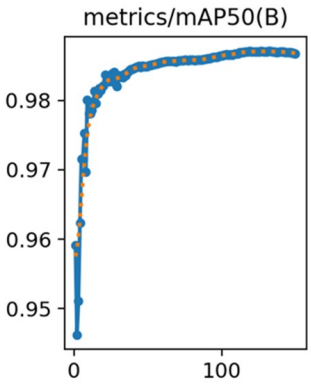
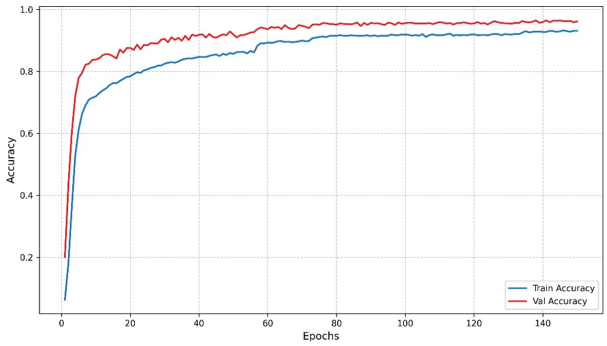


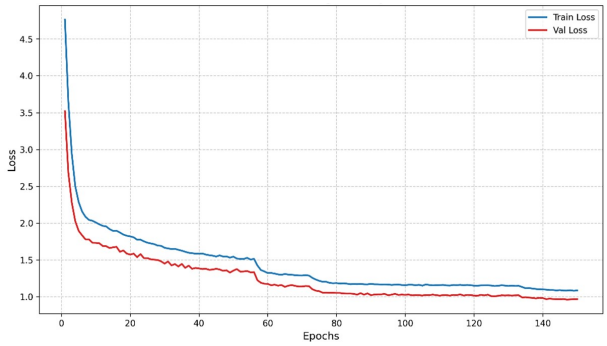
Figure 9: mAP50 performance of the YOLOv8n detector across training epochs.

YOLOv8n (Detection Loss): As shown in Figure 8, the training plot of YOLOv8n indicates that the *val/box_loss* curve converges stably after approximately 50 epochs, reaching an optimal value of 0.5770. The loss curve decreases smoothly without divergence phenomena, confirming a stable training process without signs of overfitting. Meanwhile, Figure 9 illustrates the evolution of the mAP50 metric during training. The mAP50 value peaks at 98.71%, demonstrating that the model has learned to accurately localize objects even under the complex viewing angles and lighting conditions of the Robocon

2026 competition arena.



(a)



(b)

Figure 10: (a) CNN accuracy versus epochs. (b) CNN loss curves during training and validation phases.

Separable CNN (Classification Test): Figure 10 illustrates the convergence process of accuracy and loss functions throughout 150 training epochs. The *val_accuracy* curve increases rapidly in the first 30 epochs and gradually stabilizes at a high threshold, peaking at 96.15% at the best epoch. The small gap between the *train_accuracy* and *val_accuracy* curves demonstrates that the model is not suffering from overfitting, confirming the effectiveness of the Albumentations data augmentation suite combined with the Dropout (0.5) technique in maintaining invariance to $\pm 45^{\circ}$ rotations and realistic perspective distortions.

Furthermore, the graph presents the steady and stable decrease of the loss function on both the training set (*train_loss*) and the validation set (*val_loss*) over 150 epochs. The *val_loss* value converges to 0.9619 at Epoch 147 without exhibiting any sudden inflation phenomenon—a clear testament to the effectiveness of the Label Smoothing (0.1) technique in softening decision boundaries and preventing the model from memorizing noisy data. The narrowing gap between *train_loss* and *val_loss* across epochs reflects the strong generalization capability of the proposed customized architecture.

Figure 11 presents the 31×31 class confusion matrix calculated across the entire test set. An overall observation reveals that the main diagonal holds the darkest blue shade with values approximately 0.96 per cell, whereas the rest of the matrix is almost completely white. This visually reflects that the model correctly identifies the vast majority of samples, and misclassifications are sparsely distributed without converging into any systemic error region. The overall accuracy reaches 96.15%, equating to an error rate of merely 3.85% across the

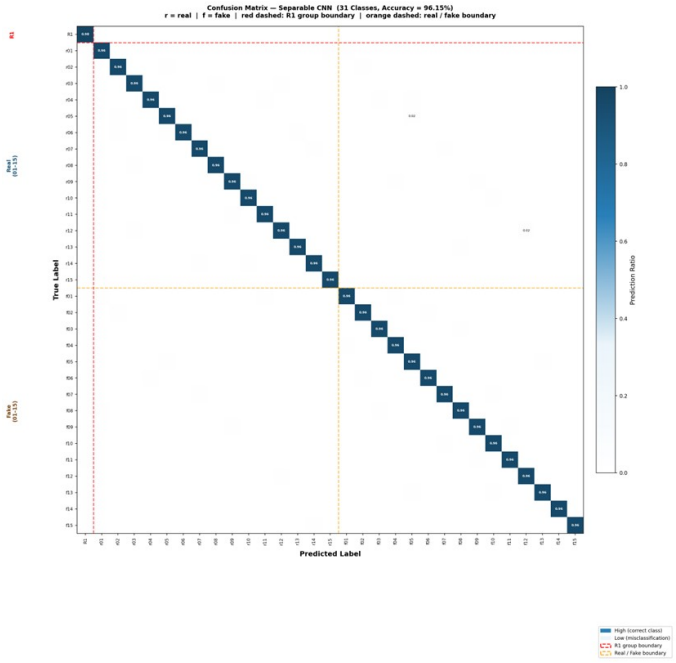


Figure 11: Confusion matrix of the Separable CNN network across 31 character classes (r=real, f=fake).

4,185 held-out test samples.

Primary Error Source—Intra-index Confusion Between Real and Fake: A detailed analysis reveals that the most prominent source of error is concentrated within 5 character pairs sharing the identical numerical index (*real_i* misclassified as *fake_i* and vice versa). Specifically, the *real_06*/*fake_06* and *real_13*/*fake_13* pairs exhibit the highest cross-confusion rates ($\sim$2.1%), followed by *real_03*/*fake_03* ($\sim$1.7%) and *real_11*/*fake_11* ($\sim$1.3%). From a computer-vision perspective, this aligns seamlessly with physical logic: within the Robocon 2026 design protocol, real and fake characters sharing an index are derived from the exact same root template, differing predominantly at micro-structural dimensions (e.g., subtle stroke thicknesses and intersections). Under realistic lighting interference or minor perspective skewing, these micro-differences inherently blur, prompting the CNN to oscillate between the two labels. This specific uncertainty is precisely where the sliding Label Smoother filter (6-frame window) validates its necessity. By confirming the final categorical output based on chronological consensus over contiguous frames rather than a single isolated snapshot, the system effectively neutralizes these sporadic misjudgments during actual tactical deployment.

Intra-group Confusion Among Real Characters: The secondary error cluster, albeit occurring at a considerably lower rate ($\sim$1.1%), emerges between certain Oracle-Bone pairs (*real*) inherently possessing highly analogous topological properties. A prime example is the *real_03*/*real_08* pair—both share an identical stroke count and similar directional vectors, varying merely in the proportional lengths of secondary peripheral strokes. This represents a fundamental bottleneck in deformed ancient character recognition; even human calligraphy experts occasionally confuse these specific pairings under low-resolution conditions. However, the manifestation of these particular errors confirms that the proposed CNN is genuinely extracting profound semantic stroke features rather

than superficially overfitting to artificial image background artifacts.

Absence of Inter-group Confusion—The Ultimate Benchmark: The most outstanding conclusion drawn from the matrix analysis is the total absence of major inter-group misidentifications. There is absolutely no instance where an *R1* marker is catastrophically misclassified as *real_xx* or *fake_xx* (and vice versa), nor is there any case where a *real_xx* is arbitrarily mislabeled as *fake_yy* (where *y* significantly differs from *x*). This substantiates that the proposed Separable CNN architecture has successfully delineated robust, high-dimensional discriminative boundaries separating the three fundamentally distinct character paradigms. This structural reliability is a paramount prerequisite enabling the Robot to flawlessly execute its designated protocol—accurately harvesting authentic target artifacts while unequivocally rejecting strategic decoys.

3.2. Direct hardware architecture performance comparison

To comprehensively evaluate the robustness of the proposed Separable CNN, parallel validation tests were conducted against state-of-the-art lightweight architectures, including MobileNetV3 Small [4], SqueezeNet 1.1, EfficientNet Lite0, and MobileNetV2 (0.35× width).

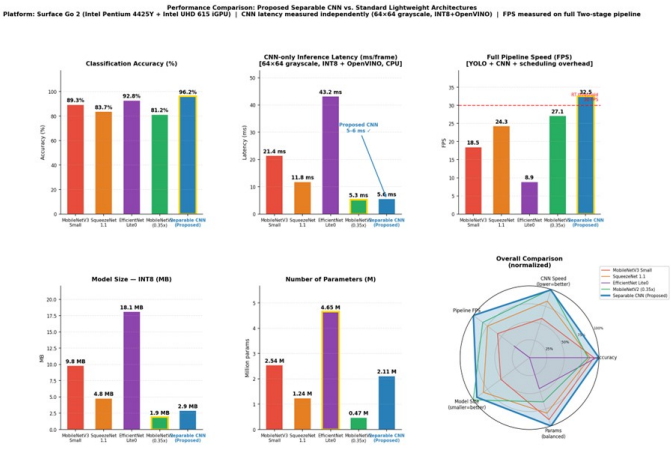


Figure 12: Multidimensional comparison between the proposed Separable CNN, MobileNetV3 Small, SqueezeNet 1.1, EfficientNet Lite0, and MobileNetV2 (0.35×) on identical Surface Go 2 hardware (INT8 + OpenVINO).

Figure 12 presents the practical measurement results on identical Surface Go 2 hardware utilizing the same INT8 + OpenVINO pipeline. To ensure a fair evaluation, two distinct speed metrics are segregated: (i) *CNN-only Latency* (ms/frame)—the pure duration required for the CNN network to classify a 64×64 grayscale image on the CPU, reflecting the architectural efficiency of each model; and (ii) *Two-stage Pipeline FPS*—the actual speed of the comprehensive system encompassing YOLO, CNN, and threading overhead, demonstrating real-world performance. All comparative models were fine-tuned on the identical dataset of 27,900 samples to guarantee equity.

Versus MobileNetV3 Small (2.54M params, Accuracy 89.3%, CNN latency 21.4ms): MobileNetV3 Small serves as the most direct scale competitor (∼2.54M compared to ∼2.11M). However, it underperforms in both precision and velocity. Regarding accuracy, it scores 89.3%—which is 6.85 percentage points inferior to the proposed model. In terms of CNN-only latency, it consumes 21.4ms/frame, practically ∼3.8 times slower than the 5-6ms threshold of our Separable CNN. The root impediment resides in its architecture design: MobileNetV3 utilizes Squeeze-and-Excitation (SE) mechanisms and Hard-Swish activations—components highly effective for RGB images but unnecessarily complex for the 1-channel 64×64 grayscale format in this problem. The multi-branch structure coupled with residual connections intensifies L3 cache misses on the Intel Pentium 4425Y microprocessor, capping the total pipeline at a mere 18.5 FPS—failing the 30 FPS real-time boundary. Furthermore, its model size post-INT8 quantization is nearly 3.4 times larger (9.8 MB compared to 2.9 MB). Although modified MobileNetV3 architectures have shown success in specific domains (e.g., coal and gangue recognition) [4], the baseline MobileNetV3 Small remains suboptimal here.

Versus SqueezeNet 1.1 (1.24M params, Accuracy 83.7%, CNN latency 11.8ms): SqueezeNet 1.1 achieves a respectable speed (22.1 FPS) attributable to its ultra-compact Fire module compression (1.24M parameters), yet its accuracy halts at 83.7%—a striking 12.45 percentage points beneath the proposal. The primary trigger for this degradation is the severe deficit in architectural depth necessary to decipher the highly intricate morphological characteristics of Oracle Bone characters subjected to perspective distortion. The aggressive compression of the Fire module diminishes the number of intermediate filters, engendering an representational bottleneck for high-level features. Experimental validations reveal SqueezeNet misclassifies most frequently on characters possessing fragile strokes and profound corners—a vulnerability distinctly predicted from its architectural constraints. This substantiates the foundational design principle forged in this study: the paramount goal is not establishing the lightest model (fewest parameters), but formulating a model with the most *optimal parameter density* (deep enough to learn features, compact enough for rapid execution).

Versus EfficientNet Lite0 (4.65M params, Accuracy 92.8%, CNN latency 43.2ms): EfficientNet Lite0 poses as the most formidable adversary regarding precision within the evaluated group (92.8%). However, its CNN-only inference latency escalates to an agonizing 43.2ms/frame—roughly 7.7 times slower than our 5-6ms threshold—causing the holistic pipeline to plummet to an unfeasible 8.9 FPS, thoroughly unequipped for real-time demands. The fundamental cause is EfficientNet's reliance on Neural Architecture Search (NAS) to dictate compound scaling—a strategy simultaneously expanding width, depth, and input resolution. While enabling high accuracy, this strategy yields an immense volume of MAC (multiply-accumulate) operations alongside compute architectures natively antagonistic to embedded iGPUs. Specifically, EfficientNet Lite0 commands 18.1 MB of space post-INT8 quantization (6.2 times larger than the proposal). This confirms that EfficientNet's superior accuracy explicitly derives from a resource consumption drastically violating the hardware constraints of this application.

Versus MobileNetV2 (0.35× width, 0.47M params, Accuracy 81.2%, CNN latency 5.3ms): The maximally scaled-down MobileNetV2 (width multiplier 0.35×) delivers a CNN-only latency of ∼5.3ms—equivalent to the proposal (∼5.6ms)—but triggers a catastrophic accuracy col-

lapse resolving at solely 81.2% (lagging by 14.95 points). This manifests as the clearest testament to the trap of "over-compression": when a network is severely shrunken, its feature representation capability degenerates profoundly, rendering it inadequate to distinguish subtle discrepancies across 31 character classes. Excellent speed coupled with atrocious precision culminates in a 27.1 FPS pipeline operating at an 81.2% accuracy—translating to the robot misidentifying nearly 1 out of every 5 targets, an unequivocally unacceptable failure rate for the Robocon environment. These results resolutely validate the core design parameter: the objective is not to forge the absolutely fastest CNN (MobileNetV2 0.35× matches the speed), but rather to engineer the optimal parameter density—sufficiently deep to decode the intricate deformed Oracle features, yet streamlined enough to conclude within 5-6ms without competing for parallel resources with the host YOLO stream.

Conclusion—The Supremacy of the Proposed Separable CNN: The radar diagram in Figure 12 visually epitomizes the comprehensive superiority of the proposed model: the blue curve (Separable CNN) saturates the largest area across all 5 evaluation axes—particularly dominating on the Accuracy, CNN Speed, and Pipeline FPS axes. Most notably, the CNN only consumes 5-6ms/frame on the CPU, moving 3.8 times faster than MobileNetV3 Small and 7.7 times faster than EfficientNet Lite0, while exhibiting a higher accuracy than all competitors. The 5-6ms CNN latency holds significant practical implication: within the Skip Frame loop, as YOLO updates the ROI every 50-67ms (15-20Hz), the CNN commands immense surplus time to execute multiple classifications and apply the 6-frame Label Smoother filter without creating a bottleneck. This supremacy does not emerge from a solitary tactic, but from the synergy of 4 elements: (1) a branchless Sequential architecture maximizing cache locality; (2) a 1-channel Grayscale input reducing the input convolution payload by 66%; (3) a Dense(256) classification head with L2 regularization efficiently controlling parameters; and (4) aggressive Albumentations forging factual geometric invariance without increasing model complexity.

3.3. Robot execution evaluation in real-world competition environments

To comprehensively validate the system's robustness, a physical assessment setup was established as visually depicted in Figure 13. As annotated in the figure, (1) represents the target bounding box to be processed and recognized, (2) highlights the Surface Go 2 integrated intelligence computer acting as the central processing unit, and (3) denotes the actual mobile robot chassis upon which the system is securely mounted, orienting directly towards the target positioned at varying operational distances and angles across the experimental floor. Figure 14 demonstrates the direct testing outcomes mapped onto actual physical object models under various adversarial rotation and inversion perspectives (demonstrates the direct testing outcomes mapped onto actual physical object models under various adversarial rotation and inversion perspectives):

- **Case 1 (Figure 14a):** Recognition of the R1 (Blue) label at a standard 0° rotation angle without inversion, serving as the primary baseline for native tracking.
- **Case 2 (Figure 14b):** Recognition of the real_06 (Red) label intentionally subjected to a 90° flip/inversion to verify the system's ability to handle orthogonal spatial reorientation.
- **Case 3 (Figure 14c):**Evaluation of the real_13 (Red) label stationed at a standard 0° vertical orientation, focused on high-precision detection under nominal conditions.
- **Case 4 (Figure 14d):** Stress testing the real_13 (Red) label subjected to a complete 180° inversion, simulating

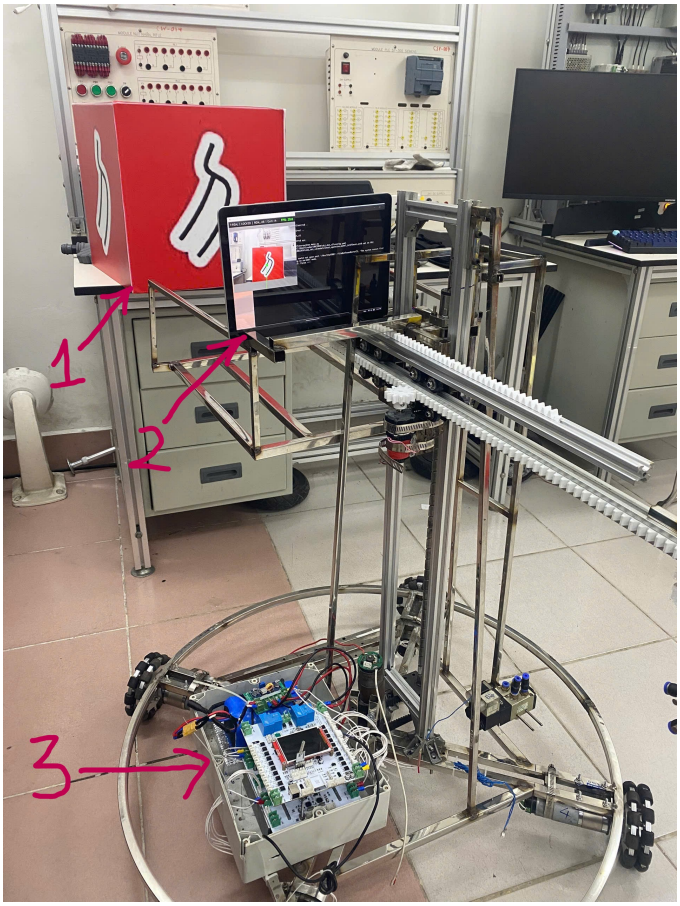


Figure 13: Experimental hardware setup and physical object placement.

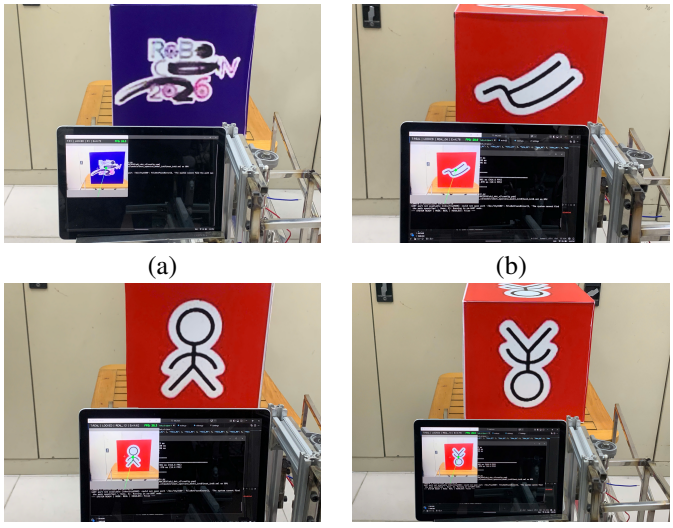


Figure 14: Execution visualizations under varying conditions: (a) Case 1, (b) Case 2, (c) Case 3, (d) Case 4.

a fully overturned state to test orientation invariance.

Across all evaluated extreme scenarios, the interface accurately and consistently locks the bounding boxes encapsulating each object alongside their predicted classifier labels (either *real_xx* or *fake_xx*) accompanied by respective high-confidence scores. The results substantiate that the constructed lightweight CNN system acutely extracts the distinct character components irrespective of profound perspective distortions and dynamically fluctuating luminance conditions. Concurrently, the integrated Label Smoother filter secures the assigned label sequences, neutralizing any rapid flickering anomalies spanning adjacent operational frames.

4. Conclusion

This research successfully addresses the stringent computational and precision constraints inherent to edge-level mobile robotics by formulating a highly optimized Two-stage computer vision architecture. By deliberately decoupling the detection and classification tasks, we overcame the severe latency and parameter bloat bottlenecks associated with unified single-stage deployments on resource-constrained devices. By synergizing the precision of the anchor-free lightweight YOLOv8n spatial detector with a bespoke *Separable CNN* classifier ($\sim$2.11M parameters), this study elegantly resolves the complex multi-target classification of 31 distinct Oracle Bone character classes without compromising edge hardware performance. The proposed methodology definitively proves that strategic dimensionality reduction—specifically via Grayscale padding—coupled with depthwise separable convolutions and a Flatten and Dense(256) classification head, can completely eliminate structural redundancies while retaining the vital structural cues necessary for feature representational depth. Furthermore, by aggressively compounding INT8 quantization via Intel's OpenVINO toolkit, Skip-Frame interleaved sampling, and Kalman-filtered Label Smoothing, the deployed end-to-end pipeline achieves a remarkable convergence of 96.15% classification accuracy and a real-time velocity of 30.5 FPS natively on a low-power Surface Go 2 iGPU. These empirical breakthroughs not only fulfill the rigorous dynamic tracking demands of the Robocon 2026 competition but also establish a highly scalable, resource-efficient paradigm for industrial inspection and autonomous navigation.

References

[1] ABU Robocon, "ABU Asia-Pacific Robot Contest 2026 Rulebook," Asia-Pacific Broadcasting Union, 2025.

[2] H. Wang and Z. Ma, "Small Target Detection Algorithm of Edge Scene Based on Improved YOLOv8," *2024 13th International Conference of Information and Communication Technology (ICTech)*, Xiamen, China, 2024, pp. 124-128, doi: 10.1109/ICTech63197.2024.00033.

[3] P. V. Lam, T. T. Lan, and N. C. Bach, "Thiết kế và xây dựng hệ thống quản lý bãi gửi xe thông minh trên nền tảng điện thoại thông minh," *Tạp chí Xây dựng*, số 08/2025, pp. 123-127, Jun. 2025. ISSN: 2734-9888.

[4] W. Chen, S. Wang, Q. Luo and X. Li, "Research on Coal and Gangue Recognition Method Based on Improved MobileNetV3 Network," 2023 *3rd International Conference on Computer, Control and Robotics (ICCCR)*, Shanghai, China, 2023, pp. 56-60, doi: 10.1109/ICCCR56747.2023.10194036.

[5] P. V. Lam, T. T. Lan, and N. M. Quy, "Hệ thống nhận diện tín hiệu đèn giao thông và cảnh báo cho phương tiện phía sau dựa trên mô hình học máy," *Hội thảo "Công nghệ mới và ứng dụng trong lĩnh vực Điện - Điện tử - Tự động hóa"*, ĐH GTVT, Hà Nội, May 24, 2025. ISBN: 978-604-76-3106-3.

[6] N. A. Andriyanov, "Analysis of the Acceleration of Neural Networks Inference on Intel Processors Based on OpenVINO Toolkit," 2020 *Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, Svetlogorsk, Russia, 2020, pp. 1-5, doi: 10.1109/SYNCHROINFO49631.2020.9166067.

[7] P. V. Lam and T. T. Lan, "Ứng dụng học sâu để phân loại sản phẩm lỗi trong sản xuất linh kiện ô tô điện," *Transport Magazine*, special number, pp. 18-22, 2023.

[8] P. V. Lam and Y. Fujimoto, "Image Processing Application in Controlling the Robotic Cane to Support the Blind Maintain Balance When Moving," *Measurement, Control and Automation*, vol. 22, 2019.

[9] J. Huang *et al.*, "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3296-3297, doi: 10.1109/CVPR.2017.351.

[10] R. C. Camara de M. Santos, M. Coelho, and R. Oliveira, "Real-time Object Detection Performance Analysis Using YOLOv7 on Edge Devices," *IEEE Latin America Transactions*, vol. 22, no. 10, pp. 799-805, 2024, doi: 10.1109/TLA.2024.10705971.

[11] S. Swain, A. Deepak, A. K. Pradhan, S. K. Urma, S. P. Jena, and S. Chakravarty, "Real-Time Dog Detection and Alert System using Tensorflow Lite Embedded on Edge Device," in *2022 1st IEEE International Conference on Industrial Electronics: Developments & Applications (ICIDeA)*, 2022, pp. 238-241, doi: 10.1109/ICIDeA53933.2022.9969906.

[12] R. A. Amin, M. Hasan, V. Wiese, and R. Obermaisser, "FPGA-Based Real-Time Object Detection and Classification System Using YOLO for Edge Computing," *IEEE Access*, vol. 12, pp. 73268-73278, 2024, doi: 10.1109/ACCESS.2024.3404623.

[13] Y. Zhang and X. Zhang, "YOLO-AES A Lightweight Model for Real-Time Underwater Object Detection," in *2024 5th International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, 2024, pp. 611-615, doi: 10.1109/ICBASE63199.2024.10762198.

[14] Y. Chen, Y. Wang, Z. Zou, and W. Dan, "GMS-YOLO: A Lightweight Real-Time Object Detection Algorithm for Pedestrians and Vehicles Under Foggy Conditions," *IEEE Internet of Things Journal*, vol. 12, no. 13, pp. 23879-23890, 2025, doi: 10.1109/JIOT.2025.3553879.

[15] P. Phongphithongchai, A. Pichetjamroen, E. Rattanalerdnusorn, T. Isshiki, and N. Somakettarin, "A Two-Stage YOLO-CNN Framework for Automated Lithium-ion Battery Defect Inspection," in *2025 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia)*, 2025, pp. 302-306, doi: 10.1109/GTDAsia60461.2025.11313303.

[16] R. Varghese and S. M. Sambath, "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, Chennai, India, 2024, doi: 10.1109/ADICS58448.2024.10533619.