Real-Time Fine-Grained Tangerine Classification via Knowledge Distillation of an Optimized YOLOv7 Model on STM32H743 Microcontroller

Farhad Hassanzadeh[1] and Sara Ershadi-Nasab[2]*

[1,2*]Computer Engineering Department, Ferdowsi University of Mashhad.

*Corresponding author(s). E-mail(s): ershadinasab@um.ac.ir.

[1,2]These authors contributed equally to this work.

Abstract

Traditional citrus fruit classification relied on weight-based sorting and manual inspection, often leading to inconsistencies in quality assessment. This research presents an optimized deep learning approach for classifying ripe, semi-ripe, and unripe tangerines using an enhanced YOLOv7-tiny convolutional object detection model. We integrate a custom-designed convolutional block attention module (CBAM) to improve feature extraction and classification performance. This proposed CBAM-enhanced model serves as the teacher in a knowledge distillation framework, guiding a compact student model to achieve high accuracy with significantly reduced computational demands.

The distillation process employs a composite loss function that includes Kullback–Leibler (KL) divergence to align soft probability distributions, mean squared error (MSE) to match intermediate feature maps, and cross-entropy for hard-label supervision. To further optimize the model, structured pruning removes redundant filters, and post-training quantization (PTQ) converts weights and activations from 32-bit floating-point to 8-bit integers, enabling deployment on resource-limited microcontrollers.

The final student model (with CBAM) achieves a classification accuracy of 95.97%, with only a 0.93% drop in accuracy compared to the original YOLOv7- tiny model with accuracy (96.9%). Moreover, the model size is reduced by 31×(from 11,945 KB to 385 KB), and inference latency decreases by 5.25×(from 27.3 ms to 5.2 ms), making it well-suitable for real-time embedded inference. The resulting student model (without CBAM) is trained on a tangerine orchard dataset

and deployed on an STM32H743 ARM Cortex-M7 microcontroller, equipped with 2 MB flash memory, a 480 MHz processor, and a DCMI interface for real-time image processing.

Keywords: Classification, Convolutional block attention module, ONNX, STM32H743, YOLOv7-tiny

1 Introduction

Fine-grained fruit classification utilizes machine learning techniques to categorize fruits into specific classes based on visual characteristics like shape, color, texture, and size. Its aim is to automate and enhance the accuracy of fruit sorting and identification processes, benefiting agriculture and food industries. Applications include quality control, inventory management, retail, and food processing. Machine learning algorithms, especially convolutional neural networks (CNNs), are commonly employed for this task, requiring large, labeled datasets. Traditional fruit sorting methods using embedded systems relied on basic machine vision techniques [1,2] but modern approaches integrate advanced machine learning techniques, like deep learning [3–6], to achieve more precise sorting outcomes, leveraging the increased computational power of embedded hardware.

In the past, researchers utilized feature extraction techniques like local binary patterns [7], histogram of oriented gradients [1], and speeded-up robust features [8], alongside machine learning algorithms such as support vector machine (SVM) [2] and k-nearest neighbors (KNN) [9], for fruit ripeness classification [1,2,10–12]. However, these methods often fell short of achieving the required accuracy.

Currently, researchers are turning to deep learning for the precise classification of seeds, leveraging their ability to extract and learn more representative image features [5] suitable for classification tasks [13–15]. Many researchers tried to implement classification methods on embedded systems. Although there are still lots of practical challenges. The research gap in fine-grained fruit classification and sorting on embedded systems lies in translating research findings into practical applications. Despite many studies exploring classification techniques for fruit sorting, challenges

hinder seamless integration. Key gaps include scalability for high throughput, robustness to variability in fruit appearance, hardware constraints on embedded systems, integration challenges between hardware and software, and adaptability to changing conditions. Bridging these gaps requires interdisciplinary collaboration and holistic solutions addressing algorithmic advancements and practical implementation challenges.

In this paper, we present a real-time embedded system for classifying the ripening level of tangerines using an ARM Cortex-M7 microcontroller with limited processing resources and memory. The core of the system is based on a significantly optimized version of the YOLOv7-tiny object detection model. Our approach focuses on drastically reducing model size and inference latency while maintaining high classification accuracy, making it suitable for low-power edge artificial intelligence (AI) applications in agriculture.

To achieve this, we first enhance the YOLOv7-tiny architecture by incorporating a convolutional block attention module (CBAM), which significantly improves the model's feature representation capability. This CBAM-enhanced model serves as the teacher in a knowledge distillation framework, where a smaller student model is trained to mimic the teacher's behavior. Next, structured pruning is applied to remove redundant filters and channels, followed by post-training quantization to convert 32- bit floating-point weights and activations to 8-bit integers. These steps collectively lead to a 31 × reduction in model size, a 5.25 × decrease in latency, and a final accuracy of 95.97%, with only a 0.93% drop compared to the original YOLOv7-tiny model.

The trained model is converted to ONNX format and deployed on the STM32H743VIT6 microcontroller using STM32CubeMX and Keil tools. Real-world performance is validated using an OV7670 camera for image capture and an OLED display for visualizing the classification output.

The main innovations of this paper are summarized as follows:

- A novel embedded AI system for automated citrus classification is developed, replacing traditional weight-based sorting and manual inspection methods. The classification targets three ripening stages—ripe, semi-ripe, and unripe—based on a custom-collected and balanced dataset. The dataset, initially consisting of 337 annotated images, is augmented

using techniques such as cropping, rotation, blurring, and denoising via Roboflow, expanding the dataset to 733 images.

- Integration of CBAM into YOLOv7-tiny, enhancing spatial and channel-wise attention to improve feature representation. The attention module improves baseline classification accuracy and acts as a high-capacity teacher model in the subsequent distillation process.
- A three-stage optimization pipeline, including (1) knowledge distillation using a hybrid loss function (KL divergence, MSE, and cross-entropy), (2) structured pruning to eliminate redundant computations, and (3) post-training quantization (PTQ) to reduce model precision. These steps result in a 31 × model size reduction (from 11,945 KB to 373 KB) and a 5.25 × improvement in latency (from 27.3 ms to 5.2 ms), with only 0.93% (from 96.9% to 95.97%) accuracy loss compared to the original YOLOv7-tiny model.
- Deployment on an STM32 ARM Cortex-M7 microcontroller, achieving real-time performance within 2 MB of flash memory. This demonstrates the practicality of deploying modern deep learning models on resource-constrained hardware for real- world agricultural applications.
- A complete end-to-end workflow for low-cost smart farming applications, from data collection and labeling to model training, compression, and embedded deployment, establishing a new benchmark for edge AI in the citrus industry.

The rest of the paper is organized as follows. Section 2 reviews related works on fruit classification using computer vision and convolutional neural networks. In Section 3, the proposed method is presented, including the block diagram of the proposed convolutional model, its internal components, and the algorithm of the proposed model. Also describes the process of dataset preparation, training the proposed convolutional model. Section 4 presenting the evaluation results of the trained model for tangerine ripeness classification and ablation study. Finally, we provide the conclusion in Section 5.

2 Related Work

2.1 A review of articles on fruit classification using convolutional networks

Zhang et al. [6] introduced a convolutional neural network architecture tailored for the fine-grained classification of banana ripening stages. The model is designed to learn a set of fine-grained image features through a data-driven mechanism. The reported accuracy of this model is 90%.

Torres et al. [5] extensively explored the use of convolutional networks in image processing, focusing specifically on fruit classification, quality control, and diagnosis. Their review outlined the latest advancements in convolutional network-based approaches until 2020, noting that 75% of commonly used models such as VGG and ResNet dominated the field, while only 25% of the research introduced novel convolutional models. Particularly noteworthy is their proposal of a basic convolutional model for fruit classification, which achieved a remarkable 100% accuracy on the 360 Fruit dataset

Alberdi et al. [16] introduced the Torchvision package, utilizing the fully connect net spinal end layer for fruit classification. While Torchvision provides a variety of pre-trained models tailored for image classification tasks, it's important to recognize that no single model is universally applicable. Users should consider factors beyond performance on the ImageNet dataset when selecting a model, such as resource constraints and computational demands. Torchvision offers twelve primary pre-trained models for image classification, including AlexNet [17], GoogLeNet [18], MobileNet [19–21], ResNet [22], VGG [23], and others, which are suitable for fine-grained inter-species data analysis.

Herman et al. [24] introduced the concept of the residual block attention mechanism. This study investigates the classification of oil palm ripeness using the residual block attention mechanism, which effectively discerns subtle differences in images. Consequently, the model achieves enhanced categorization of oil palm ripeness. The dataset comprises 400 images depicting seven stages of oil palm maturity. To augment the dataset, ten pre-processing steps, including data augmentation techniques, are applied. Ultimately, employing the DenseNet model integrated with the residual block attention mechanism yields an accuracy of 69%.

Zhu et al. [25] employed a combination of EfficientNet0-B bilinear network design and CBAM to develop the BA-EfficientNet model. This model demonstrates proficiency in automatically and precisely detecting tea oil cultivars. Through rigorous experimentation, they attained an impressive total accuracy rate of 91.7%.

Liu et al. [26] enhanced YOLOv5 for accurate citrus fruit detection and counting in orchards. They integrated a coordinated attention module to focus on fruit-dense areas and improve small-object detection. Additionally, they introduced a cross-scale two-way connection with BiFPN weighted feature aggregation to better combine multi-scale features. Using a varifocal loss function further improved training performance, achieving 95.13% accuracy.

Han et al. [27] used the original YOLOv5 model to classify cherries by quality but achieved only 78.6% accuracy after 300 iterations. To improve performance, they applied a flood filling algorithm to enhance image extraction and generate a refined dataset. Training with this new dataset rose accuracy to 99.6% after 20 sessions. The flood filling algorithm, commonly used in color-filling tools and games like Go or Minesweeper, identifies and replaces connected regions of similar features, making it effective for precise image segmentation.

Naik et al. [8] improved YOLOv5 for apple classification by replacing its activation function with Mish for smoother feature flow and adding an attention-squeezing and stimulating module to emphasize apple-specific features. The enhanced model achieved 90.6% accuracy, outperforming single shot multi box detector (+14.8%), YOLOv4 (+11.1%), and the original YOLOv5 (+3.7%).

Rajiskar et al. [28] used TensorFlow Lite with an Arduino Nano BLE to classify apples, bananas, and oranges under limited-resource conditions. They deployed a lightweight convolutional model, trained on Google Colab and implemented via Arduino IDE, achieving 93.11% accuracy.

Ruparlia et al. [29] classified tomatoes on the plant into ripe, unripe, and spoiled using CNNs. They collected diverse images from farms and online sources, applying preprocessing and data augmentation. Real-time detection was performed on an NVIDIA Jetson TX1, testing YOLOv3, YOLOv3-tiny, YOLOv4, and YOLOv4-tiny, achieving accuracies of 78.49%, 72.15%, 81.28%, and 77.25%, respectively.

Kamet et al. [30] have benchmarked modern object detection models such as YOLOv5–YOLOv7 and SSD-MobileNetv1 for multiclass fruit ripeness classification, identifying YOLOv6 as the most

accurate and efficient model for real-time quality monitoring in precision agriculture. An improved YOLOv8n-based model, YOLOv8n- GFE [31], has been proposed for nectarine detection and maturity estimation, integrating squeeze-and-excitation attention and focal distance-IoU loss to enhance recognition accuracy under challenging orchard conditions. The model achieved a 3.2% mAP improvement over the baseline YOLOv8n, demonstrating its effectiveness for real-world fruit yield assessment. A lightweight detection model, PL-YOLO [32

], was developed for accurate pomegranate identification under complex orchard conditions, introducing edge feature extraction and attention-based fusion modules to enhance feature learning and efficiency. The model achieved a 93.3% mAP and 149.5 FPS, outperforming baseline YOLO with reduced parameters and computation, making it suitable for real-time smart harvesting applications.

2.2 Comparison of Convolutional Models and Hardware used for Fruit Classification

As depicted in Table 1, most articles exhibit a common approach: they enhance the reference model and subsequently train it using the designated dataset. However, models tailored for computer deployment often exceed 40 MB in size, rendering them unsuitable for integration into embedded systems. Nevertheless, certain articles detail methods such as quantization and the utilization of TensorFlow Lite to facilitate the implementation of convolutional models on latent systems.

Analysis of Table 2 reveals that when evaluating lightweight models trained on the COCO dataset, YOLOv7-tiny demonstrates superior performance compared to other convolution models in terms of size, number of parameters, and accuracy.

2.3 Deep Learning Model Compression for Resource-Constrained Devices

Model compression techniques have become crucial for deploying deep learning models in resource-constrained environments while maintaining accuracy. Knowledge distillation, introduced by Hinton et al. [35], transfers knowledge from large ensembles to smaller models

using soft output distributions. Specialist models further refine this approach by independently handling fine-grained class distinctions.

Pruning techniques have been widely explored for compression. Network trimming by Hu et al. [36] prunes low-activation neurons, iteratively refining model efficiency. Channel pruning by He et al. [37] uses LASSO regression to remove insignificant channels, achieving a 5× speed-up with minimal accuracy loss. Network slimming by Liu et al. [38] enforces channel sparsity during training, reducing model size by 20× while maintaining accuracy.

Han et al. [39] introduced a three-step weight pruning framework that cuts storage and computation costs by an order of magnitude, enabling efficient deployment on embedded devices. Expanding on this, Wen et al. [40] proposed structured sparsity learning (SSL), which eliminates entire filters and channels, yielding up to 5.1× speedups on CPUs.

Automated pruning has also gained traction. NetAdapt by Yang et al. [41] iteratively adapts models for real-world hardware, achieving a 1.7× speed-up without accuracy loss. Auto machine learning for model compression (AMC) by He et al. [42] leverages reinforcement learning to automate compression. This method surpasses manually designed approaches in efficiency and accuracy.

Beyond pruning, Han et al. [43] developed an efficient inference engine (EIE), optimizing inference on compressed DNNs through connection pruning and weight sharing. EIE achieves 120× energy savings and 189× speed-ups over CPU implementations, making deep models feasible for edge devices.

Despite these advancements, a unified framework for evaluating pruning techniques remains lacking. Cheng et al. [44] address this by providing a taxonomy and comparative analysis of pruning methods, offering insights into post-training pruning, structured sparsity for hardware acceleration, and pruning strategies for vision transformers and diffusion models.

3 Proposed Method

In fruit quality classification, it is crucial to develop an approach that balances high accuracy with cost-effectiveness, enabling the deployment of fruit classification devices in real-world agricultural settings. To achieve this goal, implementing a deep learning model on a microcontroller instead of a high-power computing system is necessary. However, the limited flash

memory, RAM, and computational power of microcontrollers impose significant constraints on model size and complexity.

To address these limitations, this study proposes an optimized deep learning-based approach for tangerine classification using an enhanced version of the YOLOv7-tiny model. While YOLOv7-tiny is recognized as a compact and efficient convolutional model, its default size of 12 MB exceeds the memory constraints of embedded hardware. Given the maximum available memory of 2 MB on the ARM Cortex-M7 (STM32H743) microcontroller, it is essential to reduce the model size while preserving classification accuracy. The proposed approach involves knowledge distillation, structured pruning, and post-training quantization to achieve this goal. To mitigate accuracy degradation caused by model compression, the CBAM is integrated into the model. This module enhances the model's focus on key visual features, ensuring robust classification performance despite the reduced size.

3.1 Outline of the Proposed Method

The overall framework of the proposed fine-grained tangerine classification approach on an ARM microcontroller is depicted in Figure 1. The proposed system operates as follows: initially, images of tangerines in different stages, including ripe, semi-ripe, and unripe, are captured and preprocessed. Preprocessing steps include data augmentation techniques such as rotation, scaling, and contrast adjustments to enhance the diversity and robustness of the dataset. The dataset is then used to train the CBAM-enhanced YOLOv7-tiny convolutional model, which improves feature extraction by incorporating both spatial and channel attention mechanisms.

To optimize the model for deployment on an ARM Cortex-M7 microcontroller, knowledge distillation is employed to transfer knowledge from the CBAM-enhanced teacher model to a compact student model. The student model is trained using a combination of Kullback-Leibler (KL) divergence loss, mean squared error (MSE) loss, and cross-entropy loss to match soft probabilities, align feature representations, and ensure correct classification, respectively.

After training, structured pruning is applied to remove redundant filters based on their importance scores, further reducing computational complexity. Addition- ally, post-training quantization (PTQ) is implemented, converting the model's 32-bit floating-point parameters into an 8-bit integer representation to minimize memory usage and inference latency. Other optimization

techniques, such as dropout regularization and the use of convolutional filters with stride 2, are incorporated to balance accuracy and efficiency. The final optimized model is converted from PyTorch to ONNX format, reducing the model size to less than 2 MB, facilitating seamless deployment on the resource-constrained microcontroller.

For evaluation, images that were not included in the training dataset are used to assess the model's generalization performance. In real-world deployment, online image capture is conducted using the OV7670 camera, which captures tangerine images and transfers them to the ARM Cortex-M7 microcontroller via the DCMI communication interface. The microcontroller processes the images in real time and classifies the tangerines into ripe, semi-ripe, and unripe categories. The classification results are then displayed on an OLED screen, providing an efficient and cost-effective embedded solution for automated tangerine classification.

3.2 Dataset Preprocessing Step

In this section, the features of the dataset, including image capture and data augmentation, are described. The preprocessing stage commences with capturing images of fruits in the orchards to compile the dataset. The images are standardized to dimensions of 160×160 pixels and categorized into three classes: ripe, semi-ripe, and unripe. To improve the effectiveness of the proposed convolutional model, data augmentation techniques are employed. For labeling and preprocessing tasks, the online software Roboflow is utilized. The number of images captured for ripe, semi-ripe, and unripe tangerines is approximately equal, totaling 337. Leveraging Roboflow software, preprocessing operations such as image denoising, blurring, cropping, and rotation are performed, expanding the dataset to 733 images. Of this total, 80% (586 images) are allocated for model training, with 10% reserved for validation and 10% for testing the trained model. During the training phase, 80% of the dataset (586 images) undergoes 200 epochs of convolutional model training.

The dataset utilized in this study is sourced from orchards in Sari city, Iran. To proceed with training the model, it is essential to organize the tangerine dataset into three categories. An illustration of the dataset is provided in Figure 2 as a reference example.

3.3 Knowledge-Based Training and Model Compression

The overall knowledge distillation process, including the transfer of knowledge from the teacher model to the student model using KL divergence loss, MSE loss, and cross-entropy loss, is illustrated in Figure 3. The training process consists of two stages: teacher model training and student model compression. In the first stage, a teacher model is trained using the CBAM-enhanced YOLOv7-tiny architecture on a dataset collected from tangerine orchards. The dataset undergoes preprocessing steps, including image resizing, augmentation, and pixel normalization, to enhance model generalization. The CBAM refines feature representation by adaptively applying both channel and spatial attention mechanisms, allowing the model to focus on discriminative regions of tangerine images.

In the second stage, a knowledge distillation framework is used to train a compact student model under the supervision of the pre-trained teacher model. Knowledge distillation ensures that the student model learns from the softened probability outputs of the teacher, effectively preserving the performance of the larger model while significantly reducing its size. The objective is to match the output probability distributions of both models while maintaining classification accuracy. The distillation process is formulated as a loss function comprising multiple components:

$$L_{KL} = \tau^2 \sum_i p_t(i) \log \frac{p_t(i)}{p_s(i)} \tag{1}$$

where $p_t(i)$ and $p_s(i)$ represent the soft probability distributions of the teacher and student models, respectively, and τ is the temperature parameter that controls the smoothness of the probability distribution.

To further improve knowledge transfer, mean squared error (MSE) loss is applied to intermediate feature maps extracted from the teacher and student models:

$$L_{MSE} = \frac{1}{N} \sum_{i=1}^{N} \left\| F_t(i) - F_s(i) \right\|_2^2 \tag{2}$$

where $F_t(i)$ and $F_s(i)$ denote the feature maps from the teacher and student models, respectively, and N represents the total number of feature maps. Additionally, a standard cross-entropy loss is used to ensure the student model correctly predicts the class labels:

$$L_{CE} = -\sum_i y_i \log p_s(i) \tag{3}$$

Where y_i is the ground truth label, and $p_s(i)$ is the predicted probability of the student model is. The final loss function is a weighted combination of these three components:

$$L = \alpha L_{KL} + \beta L_{MSE} + \gamma L_{CE} \tag{4}$$

where $\alpha, \beta,$ and γ are hyperparameters controlling the contribution of each loss term.

In this work, the coefficients $\alpha, \beta,$ and γ in Eq. (4) were determined empirically through hyperparameter tuning on a validation subset (10% of the training data). Initially, each individual loss component (KL divergence, MSE, and CE) was evaluated to ensure that all terms contributed within a comparable numerical range. Then, a grid search was performed with $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$ and $\beta \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$, while γ was set to maintain $\alpha + \beta + \gamma = 1$. The temperature parameter τ for the KL divergence term was tested within $\{1, 2, 4, 8\}$, and $\tau = 4$ yielded the best performance by producing a softer probability distribution suitable for knowledge transfer. The final selected values were $\alpha = 0.60, \beta = 0.30, \gamma = 0.10$, and $\tau = 4$, which achieved the best validation performance and stable convergence. Sensitivity analysis indicated that small variations (± 0.05) in these coefficients caused less than 0.02% variation in mAP, confirming the robustness of the selected configuration.

Following knowledge distillation, structured pruning is applied to remove redundant filters and reduce the computational complexity of the model. Filters with low importance scores, determined by their L1-norm values, are eliminated to reduce model size while maintaining classification accuracy. The importance of each convolutional filter is calculated as:

$$S_j = \sum_{i=1}^{C} |W_{i,j}| \tag{5}$$

where $W_{i,j}$ represents the weights of the j-th filter across all input channels C, and S_j is the importance score of the filter.

To ensure seamless deployment on an embedded system, post-training quantization is applied to convert the model weights and activations from 32-bit floating point to 8-bit integer representation. This reduces memory footprint and computational cost while preserving model performance.

The final compressed model (without CBAM) is converted to ONNX format and deployed on an ARM Cortex-M7 microcontroller. Real-time classification is performed using an OV7670 camera, and the classification results are displayed on an OLED screen. The proposed approach successfully optimizes deep learning-based tangerine classification for edge AI applications while maintaining high accuracy and efficiency. The CBAM-enhanced model, however, contains attention modules (channel and spatial attention) that involve operations not directly supported by STM32Cube.AI.

Deploying CBAM on STM32 requires manual implementation or optimization, such as using integer arithmetic, lookup tables, and custom kernels, to approximate the attention mechanism efficiently on the microcontroller.

3.4 The Proposed Enhanced YOLOv7-tiny Model with Convolutional Block Attention Module

In this architectural design, the CBAM is integrated into the YOLOv7-tiny model to serve as the teacher model in the knowledge distillation process. The overall architecture of the proposed CBAM-enhanced YOLOv7-tiny, including the schematic representation and the dimensions of the feature map matrix, is illustrated in Figure 4. This model is devised based on insights gleaned from studies [12,45] and comprises three primary components: the backbone column, the convolutional block attention module, and the head.

The backbone column is responsible for extracting crucial features from the input image and forwarding them to the convolutional block attention module. The CBAM module introduces a lightweight yet effective feed-forward attention mechanism that generates attention maps along two axes: channel and spatial dimensions. These attention maps are applied to the backbone feature map to enhance significant feature representation while suppressing irrelevant background information. This refinement process enables the extraction of more discriminative features for tangerine classification.

In the final head section, the model predicts object locations and class labels by generating bounding boxes around detected objects. The integration of the CBAM module within the backbone of YOLOv7-tiny enhances feature extraction and classification accuracy, albeit with an increase in computational overhead. To mitigate this, the trained CBAM-enhanced YOLOv7-tiny model is utilized as the teacher model in a knowledge distillation framework, where its learned representations are transferred to a more compact student model. This ensures that the benefits of the attention mechanism are retained while reducing computational complexity for deployment on resource-constrained hardware.

By employing this strategy, the teacher model facilitates the distillation of rich feature representations to the student model, which undergoes further compression through pruning and quantization. The resulting student model maintains high classification performance while achieving efficient real-time inference on embedded devices.

To train the proposed convolutional model, the input image with dimensions of 640×640 is input into the network, resulting in three distinct scales of feature maps. These feature maps aid in object detection, with smaller scales suitable for detecting larger targets and larger scales adept at detecting smaller targets. Notably, the attention module within the convolutional block encompasses a multi-layer perceptron (MLP) style network in its channel part. During backpropagation, the network weights of this module are updated, thereby enhancing the accuracy of the proposed convolutional model. Subsequently, the internal components of the proposed model will be elucidated. The internal components of the Convolutional-BatchNorm-LeakyReLU (CBL) module are depicted in Figure 5.

The module begins with a convolutional filter, followed by batch normalization technique employed to enhance the performance and stability of neural networks. Lastly, the Leaky ReLU activation function is applied. Leaky ReLU behaves differently based on input values: for inputs greater than zero, it outputs the same value, while for inputs smaller than zero, the input is multiplied by 0.1. Unlike ReLU, Leaky ReLU preserves inputs less than zero, considering them with a factor of 0.1 for negative values. This ensures that none of the network nodes are entirely deactivated.

The efficient layer aggregation network (ELAN) is a computational block that influences both speed and accuracy through the analysis of factors such as memory access, I/O channel ratio, activations, and pathways. Figure 6 illustrates the internal components of ELAN. Its prominent feature lies in its time-saving capabilities, enabling more efficient performance of object detection tasks. Moreover, ELAN facilitates model execution on low-end or edge devices with constrained hardware resources.

To enhance image classification, it is positioned within the head section, comprising three integrations with filter sizes of 9×9, 13×13, and 5×5 in parallel with the input matrix, ultimately connected at the end. Figure 7 illustrates the internal components of the spatial pyramid (SP) module.

3.5 Proposed Convolutional Block Attention Module

The convolutional block attention module (CBAM) consists of two sequential sub-modules: channel attention and spatial attention. Figure 8 illustrates the schematic of CBAM applied to a feature map.

In this study, the input feature map to the leftmost CBAM module has a spatial size of 40×40 and a depth of 8 channels. First, both maximum and average pooling operations are applied along the spatial dimensions, producing two 1×1 feature maps with 8 channels each. These maps are then passed through an MLP network,

followed by concatenation and sigmoid activation. The resulting output constitutes the feature map refined by the channel attention module, as described in Eq. (6).

$$M_C(F)=\sigma(MLP(AvgPool(F))+MLP(MaxPool(F)))=\sigma(w_1w_0F^c_{avg}+w_1w_0F^c_{\max}) \quad (6)$$

In the subsequent stage, a convolutional operation is performed between the initial feature map and the output from the channel attention module. This outcome is then utilized in the spatial attention module. Within this module, separate max and average pooling operations, employing a 1×1 kernel and a stride of 1, are conducted on the layers of the feature map. Consequently, two layers with dimensions of 40×40 are generated. Following the concatenation operation, a convolutional filter with dimensions of 7×7 and padding 3 is applied, succeeded by the sigmoid

activation function. At this juncture, the output of the spatial attention module, sized 40×40, is attained, as delineated in Eq. (7).

$$M_S(F)=\sigma(Conv^{7\times7}([AvgPool(F),MaxPool(F)]))=\sigma(Conv^{7\times7}([F^S_{avg},F^S_{\max}])) \quad (7)$$

In which M_S indicates the spatial attention module. Next, the convolution operation is performed between the output of the spatial attention module (as specified in Eq. (7)) and the input feature map of the spatial attention module (as defined in Eq. (8)). Consequently, the comprehensive output of the CBAM module (as described in Eq. (9)) is achieved, possessing dimensions of 40×40 and 8 layers. While $\otimes$ indicate the convolution operation and MC indicate the channel attention module.

$$F^{'}=M_C(F)\otimes F \quad (8)$$

$$F^{''}=M_S(F^{'})\otimes F^{'} \quad (9)$$

4 Evaluation of the Proposed Method

To comprehensively evaluate the proposed optimization pipeline, four versions of the YOLOv7-tiny model were trained and analyzed: (1) the original YOLOv7-tiny model, (2) YOLOv7-tiny enhanced with CBAM, (3) a size-reduced student model derived by reducing the number of filters in the convolutional layers from 64 to 32, and (4) the student model further enhanced with CBAM. The filter reduction effectively decreases the model's size and computation, making it suitable for edge deployment on micro- controllers. However, this compression may impact classification accuracy. Therefore, the inclusion of CBAM in the student model aims to recover potential accuracy loss by enhancing the model's attention to important spatial and channel-wise features. This set of experiments enables a thorough analysis of the trade-offs between model size, computational cost, and accuracy, ensuring an optimal balance for real-world deployment. The training process was performed using Google Colab with a dataset collected from tangerine orchards.

The teacher model was designed by integrating the CBAM module into YOLOv7- tiny and training it on the tangerine dataset. To achieve an efficient student model, knowledge distillation was applied, transferring the knowledge from the teacher model to a compact version of YOLOv7-tiny.

The distillation process incorporated Kullback-Leibler (KL) divergence loss to align soft probability distributions, mean squared error (MSE) loss to match feature representations, and cross-entropy loss for supervised learning. After distillation, structured pruning was applied to remove redundant filters, further reducing computational overhead while maintaining performance. Additionally, post-training quantization (PTQ) was performed, converting weights to 8-bit integer precision to ensure seamless deployment on an embedded ARM Cortex-M7 microcontroller.

As summarized in Table 3, the original YOLOv7-tiny model is approximately 31 times larger than the final reduced-size student model. After applying knowledge distillation, pruning, and post-training quantization techniques, the student model achieved a compact size of 373 kB, representing a 4.28% decrease in accuracy (measured by mAP 0.5) compared to the original YOLOv7-tiny model. Incorporating the CBAM module into the student model further improved accuracy by 3.35%, effectively narrowing the performance gap with the teacher model to just 0.93%.

The addition of the CBAM module introduced an overhead of approximately 12 kB, which corresponds to about 3% of the size of the reduced student model. Considering the 3.35% improvement in accuracy for such a small increase in memory footprint, this enhancement demonstrates a highly efficient trade-off between accuracy and model size, making it particularly suitable for deployment on edge devices with limited computational resources.

Figure 9a, 9b, 9c, and 9d illustrate the precision-recall (PR) curves for each of the models. The legend in these diagrams presents the classification details and evaluation metrics used. The Class section indicates the object category for classification, where all represents all images, 1 corresponds to underripe tangerines, 2 represents semi- ripe tangerines, and 3 corresponds to ripe tangerines.

The evaluation metrics include precision (P), which measures the success rate of correctly detecting an object in an image, and recall (R), which assesses the actual detection rate of objects present in the image. Additionally, the mean average precision at an intersection over union (IoU) threshold of 50 percent, denoted as mAP 0.5, is used to evaluate the accuracy of detection when at least half of the predicted bounding box overlaps with the ground truth. Finally, the mAP 0.5:0.95

metric calculates the mean average precision over a range of IoU thresholds from 50 percent to 95 percent, providing a more rigorous evaluation of detection accuracy.

This evaluation demonstrates that the proposed knowledge distillation, pruning, and quantization approach effectively reduces the model size while maintaining high classification accuracy. The resulting embedded system achieves real-time classification performance with a significantly smaller model footprint, making it highly suitable for edge AI applications in smart agriculture.

4.1 Ablation Study

Table 4 presents a comprehensive ablation study conducted on YOLOv7-tiny for the task of tangerine classification. The original YOLOv7-tiny model, serving as the teacher baseline, achieves an excellent mean Average Precision (mAP) of 96.9% at a cost of a large model size (11,945 KB) and high inference latency (27.3 ms), making it less suitable for deployment on edge devices. Enhancing the teacher model with the CBAM slightly improves accuracy to 97.6%, with a marginal increase in both size and latency.

To create a compact and deployable model, we first trained a student model with reduced filters from scratch. This initial compression reduces the model size significantly, approximately 85% smaller than the teacher—at the expense of reduced accuracy (89.3% mAP). Applying knowledge distillation to this student model boosts the mAP to 92.7%, without significantly altering the model size. Although latency slightly increases due to the added complexity introduced during training, the inference time remains notably faster than the baseline. This highlights the effectiveness of distillation in recovering performance lost through filter reduction.

When pruning is applied independently, the student model's size and latency are further reduced (to 1,374 KB and 9.5 ms, respectively), though at a cost in accuracy (88.6%). Quantization alone, using 8-bit integer precision, results in a substantial size

reduction to 462.75 KB and a latency of 6.3 ms, while maintaining a relatively high accuracy of 91.8%, indicating that quantization is highly effective for edge deployment. The combination of distillation, pruning, and quantization yields a highly optimized model with a mAP of 92.62%, a compact size of 373 KB, and an inference latency of 5.1 ms. This variant represents the best trade-

off in the non-attention-based group, achieving nearly a 30-fold compression from the original model with less than 5% drop in accuracy.

Further improvements are observed when incorporating CBAM into the student model. The CBAM-augmented student, trained from scratch, achieves a mAP of 90.5%, which increases to 94.3% with distillation. When all compression techniques are applied distillation, pruning, and quantization, the resulting model achieves a mAP of 95.97%, with a size of only 385 KB and a latency of 5.2 ms. This final configuration delivers near-teacher performance with a lightweight and efficient footprint, making it highly suitable for real-time deployment on resource-constrained edge AI systems. These findings clearly demonstrate the advantage of combining attention mechanisms with model compression techniques for achieving high-accuracy, low-latency, and memory-efficient deep learning models.

The results confirm that the combination of CBAM, knowledge distillation, pruning, and quantization effectively optimizes the YOLOv7-tiny model for edge deployment. The final student model, enhanced with CBAM and all three compression techniques, achieves a 31× reduction in model size and a 5.25× reduction in inference latency compared to the original YOLOv7-tiny, while maintaining a high accuracy of 95.97%. These improvements make the model highly suitable for real-time tangerine classification on resource-constrained devices such as microcontrollers.

Figure 10, Figure 11, and Figure 12 illustrate the results of our ablation study on the YOLOv7-tiny model, comparing the performance of variants with and without CBAM. Figure 10 presents classification accuracy (mAP 0.5), highlighting the effectiveness of each optimization technique—knowledge distillation, pruning, and quantization—individually and in combination.

Figure 11 shows the corresponding model sizes in kilobytes, demonstrating the substantial compression achieved through quantization and pruning. Finally, Figure 12 depicts inference latency, confirming that the proposed optimizations lead to significant improvements in runtime efficiency, making the models more suitable for real-time edge deployment.

5 Conclusion

In this study, we focused on reducing the size of the YOLOv7-tiny convolutional model while maintaining high accuracy for classifying ripe, semi-ripe, and unripe tangerines. To achieve this, we introduced a teacher-student learning framework, where a CBAM- enhanced YOLOv7-tiny model served as the teacher, and a compact student model was obtained through knowledge distillation, structured pruning, and post-training quantization. The final student model achieved a classification accuracy of 95.97%, with only a 0.93% decrease in mAP compared to the original YOLOv7-tiny baseline model (96.9%), while being approximately 31× smaller in model size (from 11,945 KB to 385 KB) and 5.25× faster in inference latency time (from 27.3 ms to 5.2 ms). This optimized model was successfully deployed on the STM32H743VIT6 microcontroller, demonstrating the feasibility of real-time classification on resource-constrained embedded hardware.

Our objective was to develop a cost-effective and efficient tangerine classification system that adheres to tight hardware constraints, and this goal has been effectively realized. The proposed approach enables qualitative grading of harvested tangerines, making it suitable for practical applications in agricultural automation.

As part of future work, we plan to expand the dataset to include images captured under varying lighting conditions, including semi-dark and low-light environments, to improve robustness in real-world scenarios. Moreover, future research may explore classifying tangerines directly on the tree by incorporating domain adaptation techniques. Another promising direction is the integration of alternative attention mechanisms, such as squeeze-and-excitation (SE) modules, to further enhance model efficiency and accuracy.

In this work, the compressed student model without the CBAM module was successfully deployed on an STM32H743 ARM Cortex-M7 microcontroller using the STM32Cube. AI and CubeMX toolchain. The deployed version includes all essential layers—convolutional, batch normalization, activation (LeakyReLU), pooling, and fully connected—supported by the STM32Cube.AI operator library. However, the CBAM-enhanced variant could not yet be fully implemented because of the current STM32Cube. AI framework lacks native support for custom attention operations such as global pooling across multiple branches and element-wise channel weighting. Despite this limitation, the deployed student model achieved real-time inference with a latency of

5.2 ms and an accuracy of 92.62%, validating the efficiency of the proposed compression pipeline (distillation, pruning, and quantization) for embedded vision tasks. Future work will explore the integration of CBAM and other lightweight attention modules through custom layer implementation or model conversion to extend compatibility with STM32-based AI accelerators.

6 Conflict of Interest Declaration

The authors declare that there are no conflicts of interest associated with this manuscript.

7 Data Availability

The dataset used in this article is publicly available.

https://github.com/saraershadi/Tangerines-SariCity/

References

1. Muhathir, M., Santoso, M. H., and Muliono, R. "Analysis naïve Bayes in classifying fruit by utilizing HOG feature extraction", Journal of Informatics and Telecommunication Engineering, **4**(1), pp. 151–160 (2020). https://doi.org/10.31289/jite.v4i1.3860

2. Muhathir, M., Hidayah, W., and Ifantiska, D. "Utilization of support vector machine and speeded up robust features extraction in classifying fruit imagery", Computer Engineering and Applications, **9**(3), pp. 183–193 (2020). https://doi.org/10.18495/COMENGAPP.V9I3.347

3. Weng, W., Lai, Z., Cui, Z., et al. "GCD-YOLO: a deep learning network for accurate tomato fruit stalks identification in unstructured environments", Smart Agricultural Technology, pp. 101465 (2025). https://doi.org/10.1016/j.atech.2025.101465

4. Bai, Y., Yu, J., Yang, S., et al. "An improved YOLO algorithm for detecting flowers and fruits on strawberry seedlings", Biosystems Engineering, **237**, pp. 1–12 (2024). https://doi.org/10.1016/j.biosystemseng.2023.11.008

5. Naranjo-Torres, J., Mora, M., Hernández-Garcia, R., et al. "A review of convolutional neural network applied to fruit image processing", Applied Sciences, **10**(10), pp. 3443 (2020). https://doi.org/10.3390/app10103443

6. Zhang, Y., Lian, J., Fan, M., et al. “Deep indicator for fine-grained classification of banana’s ripening stages”, Journal on Image and Video Processing, 2018, 46 (2018). https://doi.org/10.1186/s13640-018-0284-8

7. Song, K. C., Yan, Y. H., Chen, W. H., et al. “Research and perspective on local binary pattern”, Acta Automatica Sinica, **39**(6), pp. 730–744 (2013). https://doi.org/10.1016/s1874-1029(13)60051-8

8. Naik, S. and Patel, B. “Machine vision based fruit classification and grading - a review”, International Journal of Computer Applications, **170**(9), pp. 22–34 (2017). https://doi.org/10.5120/ijca2017914937

9. Astuti, I. F., Nuryanto, F. D., Widagdo, P. P., et al. “Oil palm fruit ripeness detection using k-nearest neighbour”, Journal of Physics: Conference Series, pp. 12028 (2019). https://doi.org/10.1088/1742-6596/1277/1/012028

10. Mustafa, N. B. A., Ahmed, S. K., Ali, Z., et al. “Agricultural produce sorting and grading using support vector machines and fuzzy logic”, International Conference on Signal and Image Processing Applications, pp. 391–396 (2009). https://doi.org/10.1109/ICSIPA.2009.5478684

11. Dubey, S. R. and Jalal, A. S. “Detection and classification of apple fruit diseases using complete local binary patterns”, International Conference on Computer and Communication Technology, pp. 346–351 (2012). https://doi.org/10.1109/ICCCT.2012.76

12. Liu, Z., Wu, J., Fu, L., et al. “Improved kiwifruit detection using pre-trained VGG16 with RGB and NIR information fusion”, IEEE Access, **8**, pp. 2327–2336 (2019). https://doi.org/10.1109/ACCESS.2019.2962513

13. Tang, Z., Fang, L., Sun, S., et al. “ML-DETR: multiscale-lite detection transformer for identification of mature cherry tomatoes”, IEEE Transactions on Instrumentation and Measurement, **74** (2025). https://doi.org/10.1109/TIM.2025.3608317

14. Zhu, X., Chen, F., Zheng, Y., et al. “Detection of camellia oleifera fruit maturity in orchards based on modified lightweight YOLO”, Computers and Electronics in Agriculture, **226**, pp. 109471 (2024). https://doi.org/10.1016/j.compag.2024.109471

15. Wang, L., Wang, S., Wang, B., et al. “Jujube-YOLO: a precise jujube fruit recognition model in unstructured environments”, Expert Systems with Applications, pp. 128530 (2025). https://doi.org/10.1016/j.eswa.2025.128530

16. Albardi, F., Kabir, H. M. D., Bhuiyan, M. M. I., et al. “A comprehensive study on torchvision pre-trained models for fine-grained inter-species classification”, IEEE International Conference on Systems, Man, and Cybernetics, pp. 2767–2774 (2021). https://doi.org/10.1109/SMC52423.2021.9659161

17. Krizhevsky, A., Sutskever, I., and Hinton, G. E. “Imagenet classification with deep convolutional neural networks”, Communications of the ACM, **60**(6), pp. 84–90 (2017). https://doi.org/10.1145/3065386

18. Szegedy, C., Liu, W., Jia, Y., et al. “Going deeper with convolutions”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015). https://doi.org/10.1109/CVPR.2015.7298594

19. Howard, A. G., Zhu, M., Chen, B., et al. “Mobilenets: efficient convolutional neural networks for mobile vision applications”, arXiv preprint arXiv:1704.04861 (2017). https://doi.org/10.48550/arXiv.1704.04861

20. Sandler, M., Howard, A., Zhu, M., et al. “Mobilenetv2: inverted residuals and linear bottlenecks”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018). https://doi.org/10.1109/CVPR.2018.00474

21. Howard, A., Sandler, M., Chu, G., et al. “Searching for mobilenetv3”, Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1314–1324 (2019). https://doi.org/10.1109/ICCV.2019.00140

22. He, K., Zhang, X., Ren, S., et al. “Deep residual learning for image recognition”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90

23. Simonyan, K. and Zisserman, A. “Very deep convolutional networks for large-scale image recognition”, arXiv preprint arXiv:1409.1556 (2015). https://doi.org/10.48550/arXiv.1409.1556

24. Herman, H., Susanto, A., Cenggoro, T. W., et al. “Oil palm fruit image ripeness classification with computer vision using deep learning and visual attention”, Journal of Telecommunication, Electronic and Computer Engineering, **12**(2), pp. 21–27 (2020). https://jtec.utem.edu.my/jtec/article/view/5543

25. Zhu, X., Yu, Y., Zheng, Y., et al. “Bilinear attention network for image-based fine-grained recognition of oil tea (camellia oleifera abel.) cultivars”, Agronomy, **12**(8), pp. 1846-1861 (2022). https://doi.org/10.3390/agronomy12081846

26. Liu, X., Li, G., Chen, W., et al. “Detection of dense citrus fruits by combining coordinated attention and cross-scale connection with weighted feature fusion”, Applied Sciences, **12**(13), pp. 6600-6617 (2022). https://doi.org/10.3390/app12136600

27. Han, W., Jiang, F., and Zhu, Z. “Detection of cherry quality using YOLOV5 model based on flood filling algorithm”, Foods, **11**(8), pp. 1127-1136 (2022). https://doi.org/10.3390/foods11081127

28. Rajasekar, L., Babu, C. G., and Sharmila, D. “Identification of fruits and vegetables using embedded sensor”, IOP Conference Series: Materials Science and Engineering, 1084(1), pp. 12095-12103 (2021). https://doi.org/10.1088/1757-899X/1084/1/012095

29. Ruparelia, S., Jethva, M., and Gajjar, R. “Real-time tomato detection, classification, and counting system using deep learning and embedded systems”, Proceedings of the International e-Conference on Intelligent Systems and Signal Processing, pp. 511–522 (2022). https://doi.org/10.1007/978-981-16-2123-9_39

30. Kamat, P., Gite, S., Chandekar, H., et al. “Multi-class fruit ripeness detection using YOLO and SSD object detection models”, Discover Applied Sciences, **7**(9), pp. 931-948 (2025). https://doi.org/10.1007/s42452-025-07617-7

31. Ji, B., Zhao, J., Tao, F., et al. “A novel nectarine fruit maturity detection and classification counting model based on YOLOv8n”, IEEE Transactions on AgriFood Electronics, 3(1), pp. 144-155 (2024). https://doi.org/10.1109/TAFE.2024.3488747

32. Chen, X. and Wang, G. “PL-YOLO: a lightweight method for real-time detection of pomegranates”, Journal of Real-Time Image Processing, **22**(4), pp. 1–13 (2025). https://doi.org/10.1007/s11554-025-01729-4

33. Xu, B., Cui, X., Ji, W., et al. “Apple grading method design and implementation for automatic grader based on improved YOLOv5”, Agriculture, **13**(1), pp. 124-142 (2023). https://doi.org/10.3390/agriculture13010124

34. Assunção, E., Gaspar, P. D., Alibabaei, K., et al. “Real-time image detection for edge devices: a peach fruit detection application”, Future Internet, **14**(11), pp. 323-335 (2022). https://doi.org/10.3390/fi14110323

35. Hinton, G., Vinyals, O., and Dean, J. “Distilling the knowledge in a neural network”, arXiv preprint arXiv:1503.02531 (2015). https://doi.org/10.48550/arXiv.1503.02531

36. Hu, H., Peng, R., Tai, Y.-W., et al. “Network trimming: a data-driven neuron pruning approach towards efficient deep architectures”, arXiv preprint arXiv:1607.03250 (2016). https://doi.org/10.48550/arXiv.1607.03250

37. He, Y., Zhang, X., and Sun, J. “Channel pruning for accelerating very deep neural networks”, Proceedings of the IEEE International Conference on Computer Vision, pp. 1389–1397 (2017). https://doi.org/10.1109/ICCV.2017.155

38. Liu, Z., Li, J., Shen, Z., et al. “Learning efficient convolutional networks through network slimming”, Proceedings of the IEEE International Conference on Computer Vision, pp. 2736–2744 (2017). https://doi.org/10.1109/ICCV.2017.298

39. Han, S., Pool, J., Tran, J., et al. “Learning both weights and connections for efficient neural networks”, arXiv preprint arXiv:1506.02626 (2015). https://doi.org/10.48550/arXiv.1506.02626

40. Wen, W., Wu, C., Wang, Y., et al. “Learning structured sparsity in deep neural networks”, arXiv preprint arXiv:1608.03665 (2016). https://doi.org/10.48550/arXiv.1608.03665

41. Yang, T.-J., Howard, A., Chen, B., et al. “Netadapt: platform aware neural network adaptation for mobile applications”, Proceedings of the European Conference on Computer Vision, pp. 285–300 (2018). https://doi.org/10.1007/978-3-030-01249-6_18

42. He, Y., Lin, J., Liu, Z., et al. “Amc: automl for model compression and acceleration on mobile devices”, Proceedings of the European Conference on Computer Vision, pp. 784–800 (2018). https://doi.org/10.1007/978-3-030-01234-2_48

43. Han, S., Liu, X., Mao, H., et al. “EIE: efficient inference engine on compressed deep neural network”, ACM SIGARCH Computer Architecture News, **44**(3), pp. 243–254 (2016). https://doi.org/10.1145/3007787.3001163

44. Cheng, H., Zhang, M., and Shi, J. Q. “A survey on deep neural network pruning taxonomy, comparison, analysis, and recommendations”, IEEE Transactions on Pattern Analysis and Machine Intelligence, **46**(12), pp. 10558–10578 (2024). https://doi.org/10.1109/TPAMI.2024.3447085

45. Zhu, L., Geng, X., Li, Z., et al. “Improving YOLOv5 with attention mechanism for detecting boulders from planetary images”, Remote Sensing, **13**(18), pp. 3776-3795 (2021). https://doi.org/10.3390/rs13183776

Farhad Hassanzadeh received his M.Sc. degree in Computer Engineering in 2024. His research interests focus on artificial intelligence in embedded systems, particularly the implementation of machine learning and deep learning algorithms on resource-constrained hardware platforms. His work involves embedded processors and edge devices for real-time intelligent applications. His areas of interest include embedded AI, edge computing, and intelligent IoT systems.

Sara Ershadi-Nasab received her Ph.D. in Digital Systems (Artificial Intelligence) from Sharif University of Technology in 2017. She is currently an Assistant Professor in the Computer Engineering Department at Ferdowsi University of Mashhad, Iran. Her research focuses on artificial intelligence at the edge, embedded systems, FPGA-based acceleration, hardware-software co-design, and intelligent IoT applications. Her work emphasizes the efficient implementation of deep learning models on resource-constrained platforms, smart sensing systems, and real-time intelligent embedded solutions for applications in smart grids, healthcare, and edge computing.

Figures Caption

Figure 1. Outline of the proposed method of implementation.

Figure 2. Fine-grained classification of tangerines:
Unripe (left), Semi-Ripe (middle), Ripe (right).

Figure 3. Knowledge distillation process where the teacher model guides the student model using KL divergence loss, MSE loss, and cross-entropy loss.

Figure 4. Block diagram outline of the proposed CBAM-enhanced YOLOv7-tiny as the teacher model.

Figure 5. Block diagram of CBL module.

Figure 6. Block diagram of ELAN module.

Figure 7. Block diagram of SP module.

Figure 8. A summary of the proposed convolutional block attention module.

Figure 9. Precision-recall diagrams for different YOLOv7-tiny models.

Figure 10. Comparison of classification accuracy (mAP@0.5) across different models derived from YOLOv7-tiny, with and without CBAM. The ablation study highlights the individual and combined impacts of knowledge distillation, pruning, and quantization techniques for tangerine classification.

Figure 11. Comparison of model size (in KB) for various YOLOv7-tiny derivatives, with and without CBAM. The ablation study illustrates the effectiveness of model compression strategies in reducing storage requirements while preserving accuracy.

Figure 12. Comparison of inference latency (in milliseconds) for different YOLOv7- tiny variants, with and without CBAM. The results demonstrate how distillation, pruning, and quantization contribute to faster real-time performance suitable for edge deployment.

Tables Caption

Table 1. Comparison of convolutional models and hardware utilized in fine-grained fruit classification.

Table 2. Comparison of lightweight convolutional models on the COCO dataset.

Table 3. Performance Comparison of Different Model Variants.

Table 4. Ablation Study on YOLOv7-tiny for Tangerine Classification.

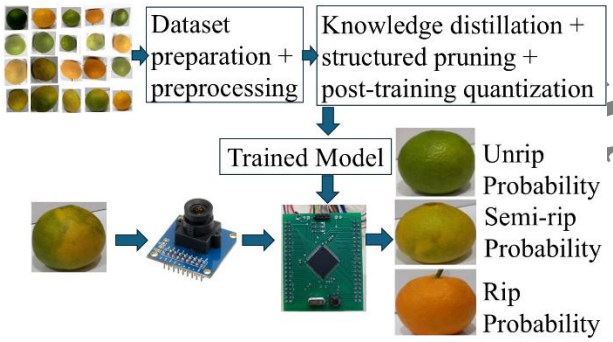

Figure 1. Outline of the proposed method of implementation.

Figure 2. Fine-grained classification of tangerines:
Unripe (left), Semi-Ripe (middle), Ripe (right).

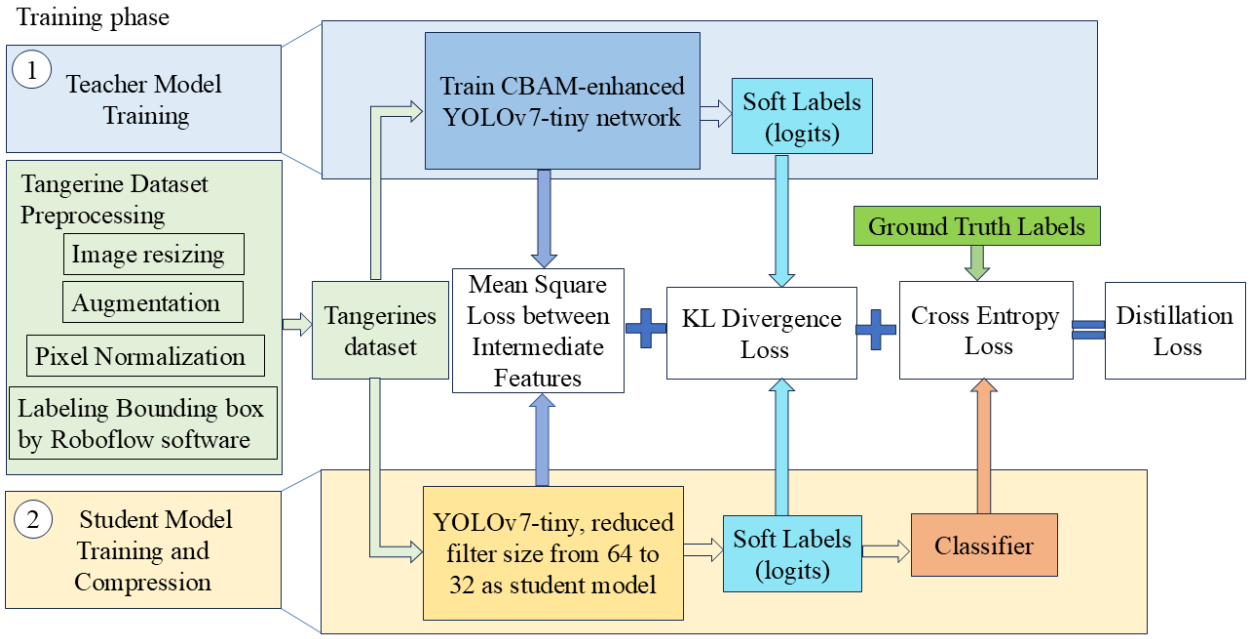

Figure 3. Knowledge distillation process where the teacher model guides the student model using KL divergence loss, MSE loss, and cross-entropy loss.

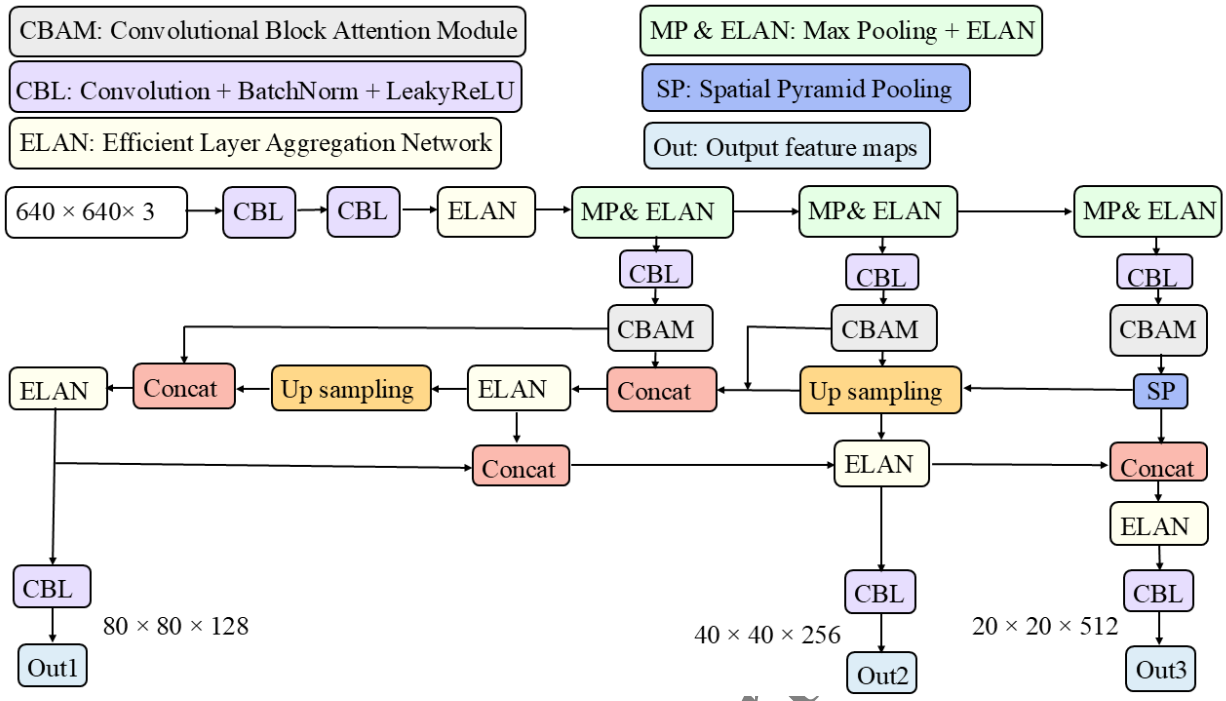

Figure 4. Block diagram outline of the proposed CBAM-enhanced YOLOv7-tiny as the teacher model.

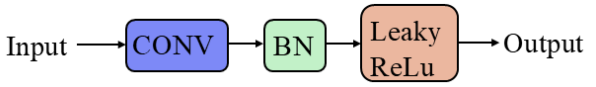

Figure 5. Block diagram of CBL module.

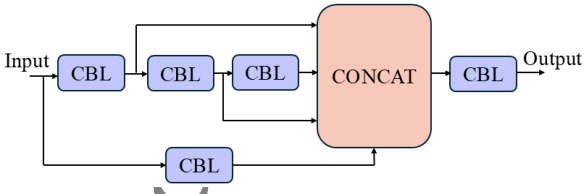

Figure 6. Block diagram of ELAN module.

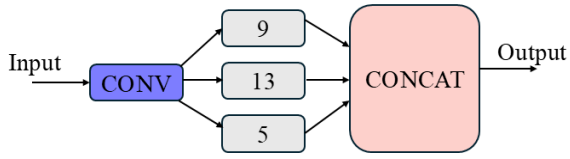

Figure 7. Block diagram of SP module.

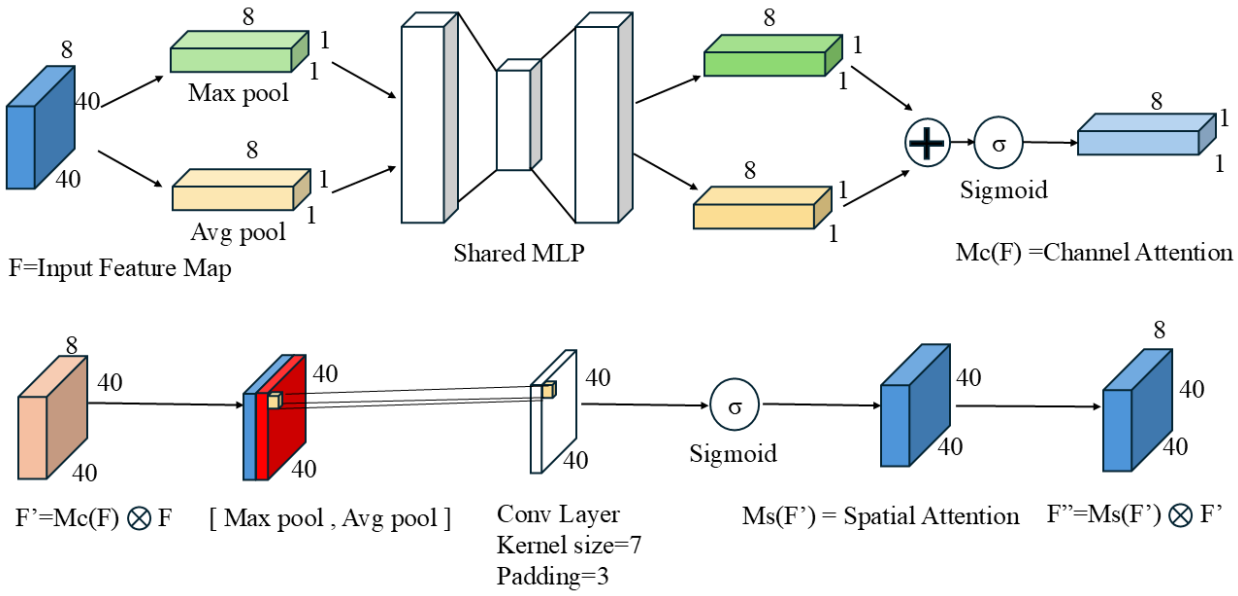

Figure 8. A summary of the proposed convolutional block attention module.

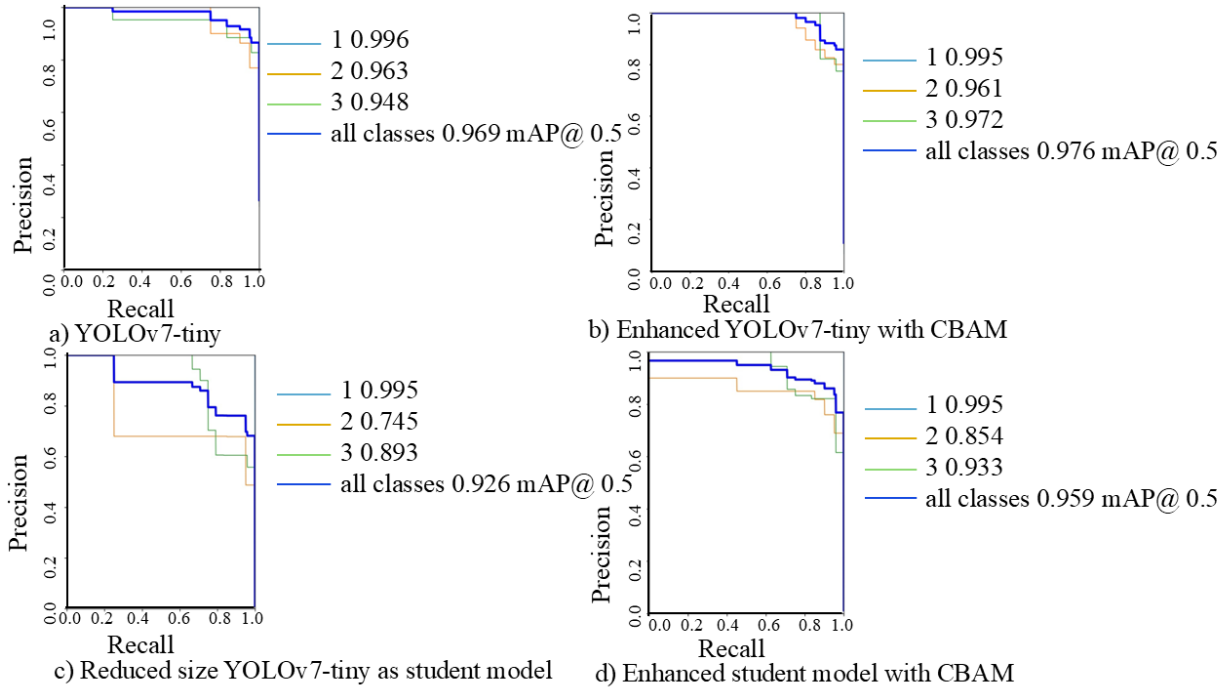

Figure 9. Precision-recall diagrams for different YOLOv7-tiny models.

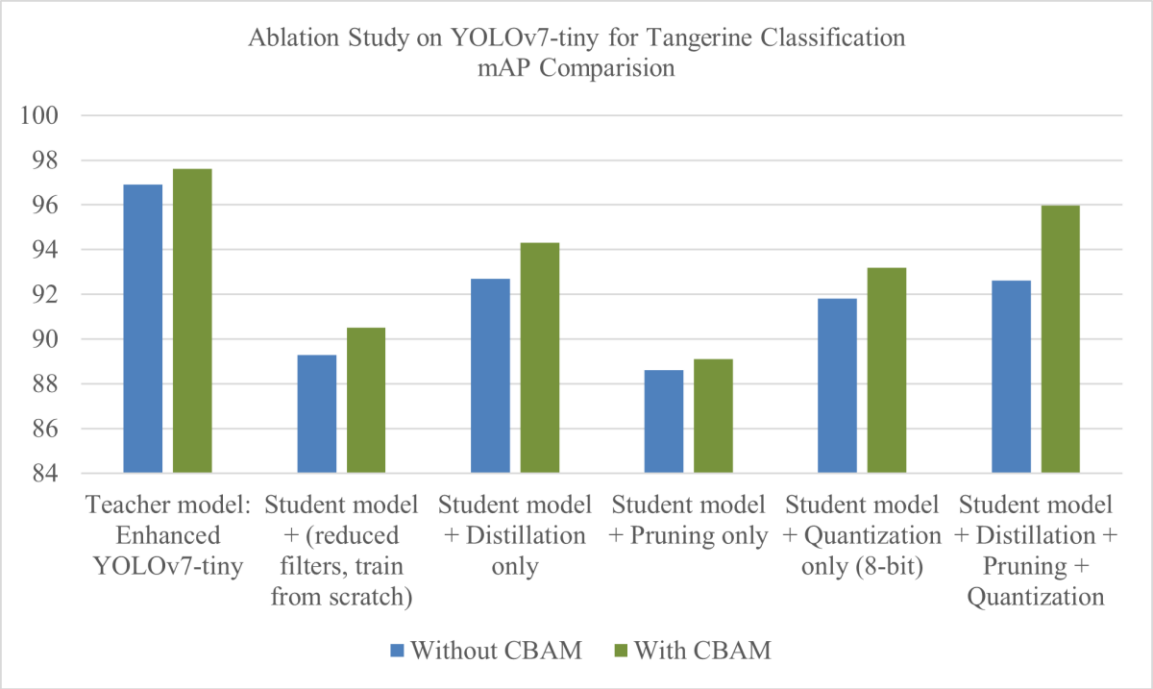

Figure 10. Comparison of classification accuracy (mAP@0.5) across different models derived from YOLOv7-tiny, with and without CBAM. The ablation study highlights the individual and combined impacts of knowledge distillation, pruning, and quantization techniques for tangerine classification.

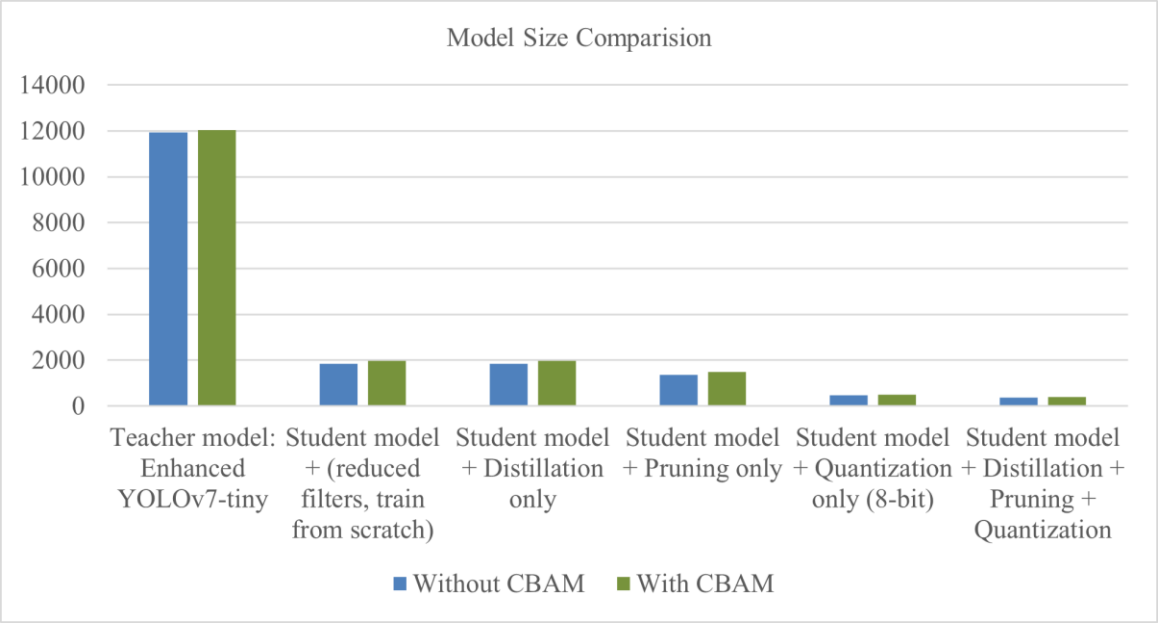

Figure 11. Comparison of model size (in KB) for various YOLOv7-tiny derivatives, with and without CBAM. The ablation study illustrates the effectiveness of model compression strategies in reducing storage requirements while preserving accuracy.

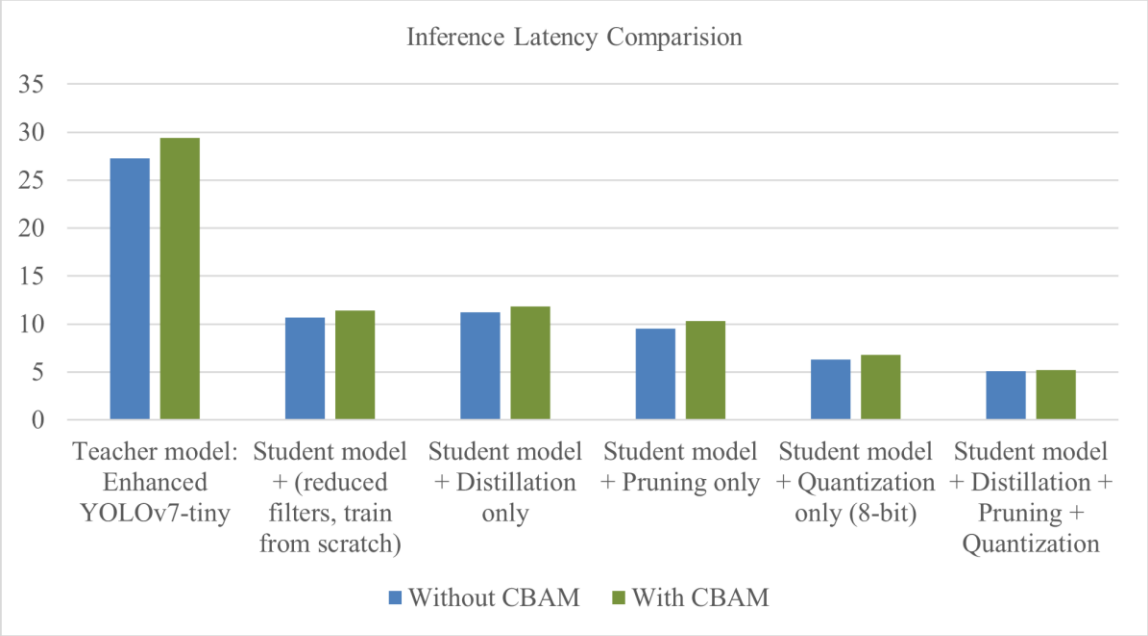

Figure 12. Comparison of inference latency (in milliseconds) for different YOLOv7- tiny variants, with and without CBAM. The results demonstrate how distillation, pruning, and quantization contribute to faster real-time performance suitable for edge deployment.

Table 1. Comparison of convolutional models and hardware utilized in fine-grained fruit classification.

CNN Model	Improvement method	Size (MB)	Hardware	AP%	Fruit
Researcher	...	...	PC	90	banana [6]
Torchvision	Spinal net	40	PC	99	fruit360 [16]
DenseNet	Residual-based attention	43.6	PC	69	oil palm [24]
B0-EfficientNet	CBAM	...	PC	91.7	oil tea [25]
YOLOv5	loss function	168	PC	95.3	citrus fruits [26]
YOLOv5	SE module	168	PC	90.6	apple [33]
YOLOv5	Flood filling	168	PC	90	cherry [27]
Researcher	Tflite	<1	ARDUINO Nano	93.11	fruits [28]
YOLOv3-tiny	...	33.2	NVIDIA Jetson TX1	72.15	tomato [29]
YOLOv4-tiny	...	23.1	NVIDIA Jetson TX1	77.25	tomato [29]
MobileNet	quantization	14	Raspberry Pi 4	88	peach [34]

Table 2. Comparison of lightweight convolutional models on the COCO dataset.

Model	YEAR	mAP	size
YOLOv7-tiny	2022	56.7	12MB
YOLOv4-tiny	2020	40.2	23.1MB
YOLOv3-tiny	2018	54	33.2MB

Table 3. Performance Comparison of Different Model Variants.

Model	mAP 0.5(%)	mAP 0.5:0.95(%)	Size (kB)
YOLOv7-tiny	96.9	79.7	11945
Enhanced YOLOv7-tiny with CBAM	97.6	81.9	12039
Student Model (Distilled & Pruned & Quantized)	92.62	73.5	373
CBAM Enhanced Student Model (Distilled & Pruned & Quantized)	95.97	78.8	385

Table 4. Ablation Study on YOLOv7-tiny for Tangerine Classification.

Model Variant	mAP 0.5	Size (KB)	Latency (ms)
Teacher model: YOLO 7-tiny (Baseline)	96.9	11,945	27.3
Student model (reduced filters, train from scratch)	89.3	1,851	10.7
Student model + Distillation only	92.7	1,850	11.2
Student model + Pruning only	88.6	1,374	9.5
Student model + Quantization only (8-bit)	91.8	462.75	6.3
Student model + Distillation + Pruning + Quantization	92.62	373	5.1
Teacher model: Enhanced YOLO 7-tiny with CBAM	97.6	12,039	29.4
Student model + CBAM (reduced filters, train from scratch)	90.5	1,966	11.4
Student model + CBAM+ Distillation only	94.3	1,965	11.8
Student model + CBAM+ Pruning only	89.1	1,486	10.3
Student model + CBAM+ Quantization only (8-bit)	93.2	491.25	6.8
Student model + CBAM+ Distillation + Pruning + Quantization	95.97	385	5.2