ORIGINAL RESEARCH **Open access**

An End-to-End Real-Time Framework for Canola Damage Detection Using a Semi-Automatic Crusher and Lightweight YOLOv5s

Fatima El Idrissi[1*], Samira Bennani[1], Youssef Benali[2], Ahmed Ben Ali[3]

Abstract

The existing technique for identifying damage in canola kernels remains inefficient and highly time-consuming. Preparing samples requires significant manual effort, while visually distinguishing sound seeds from damaged ones is often difficult due to their tiny size and subtle differences. Currently, no combined hardware-and-software system is available to speed up this vital but lengthy procedure. This research introduces a complete end-to-end damage detection system comprising two key components: a semi-automatic device that rapidly crushes canola seeds for rapid sample preparation and an accompanying object detection algorithm derived from a compressed YOLOv5s model. The compact detection model was deployed on an Edge AI platform and connected to the crusher to enable real-time damage identification. Two Convolutional Neural Network (CNN) architectures, specifically ShuffleNetV2 and MobileNetV3, were evaluated based on model parameters, computational cost, and overall model size as replacements for the standard CSPDarknet53 CNN backbone in YOLOv5s. The top-performing model was then implemented on an NVIDIA Jetson Nano embedded platform for real-time inference. Findings reveal that, with respect to model size, the ShuffleNetV2_YOLOv5s model was 56.5% and 80.4% smaller than the MobileNetV3_YOLOv5s and the baseline YOLOv5s models, respectively. In terms of computational cost, the ShuffleNetV2_YOLOv5s model was 64.1% and 99.5% less expensive than the MobileNetV3_YOLOv5s and the baseline YOLOv5s models, respectively. This research established an end-to-end framework for identifying canola damage, supported by a real-time, compact damage-detection model.

Keywords Object detection, Edge-AI, Canola quality, Real-time inferencing, Network compaction

*Correspondence:
Fatima El Idrissi
fatima.idrissi@uca.ma
[1] Department of AI for Saffron and Argan Fruit Maturation Forecasting, Faculty of Agriculture, University of Marrakech, Marrakech, Morocco
[2] Department of Smart Traditional Terraced Farm Monitoring, Faculty of Agriculture, University of Fez, Fez, Morocco
[3] Department of Machine Learning for Soil Erosion Control, Faculty of Agriculture, University of Casablanca, Casablanca, Morocco

Introduction

Canada stands as the world's leading producer and exporter of canola. The market price of canola is heavily determined by harvest quality, which depends directly on the overall count of damaged seeds. The primary forms of damage in canola involve heat- and immature-seed issues, identifiable solely by the cotyledon's visual appearance: dark brown indicates heat damage, while green indicates immature seeds. The conventional method for assessing these damages involves crushing canola seeds with a handheld roller and then having trained staff evaluate the results by direct observation.

This is an open access article distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0), which permits non-commercial use, distribution, reproduction, and adaptation in any medium or format, provided that appropriate credit is given to the original author(s) and source, a link to the license is provided, and any changes made are indicated. Adapted material must be distributed under the same license.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

These damage types significantly reduce the total value and market price of canola kernels by interfering with later stages of oil extraction and refining. Oil produced from immature seeds appears darker due to its higher chlorophyll content. Removing these unwanted pigments requires processors to use larger quantities of clay, necessitating an additional filtration stage. This process results in reduced final output and higher overall production expenses. In addition, immature seeds show poorer storage stability and contain less oil per tonne. The enzyme pheophorbide-a-oxygenase plays a key role in breaking down chlorophyll as canola seeds mature [1].

Nevertheless, during seed development, factors such as frost exposure, hot dry conditions, or excessively hot air from fans in storage interfere with enzyme activity. These enzymes function effectively only at temperatures above 5 °C and when seed moisture exceeds 20%. Such harsh conditions, either before harvesting or due to faulty silo storage, can deactivate the enzymes, halting seed maturation entirely. High temperatures also primarily cause the development of heated seeds. Although the origins of immature and heated seed formation are well understood, no methods currently exist to correct these issues.

As a result, identifying damaged canola seeds represents a critical task for grain buyers and elevators in accurately evaluating canola value before export. Canola is graded based on the overall damage percentage across three categories (i.e., 0–5%, 5–12%, 12–25%). The present sample preparation approach relies on a handheld roller. Seeds are crushed manually, and damage is assessed by visual inspection using a color reference chart. Furthermore, the conventional damage detection process is labor-intensive and slow due to the manual preparation phase, and it is also vulnerable to personal bias and human error because decisions rest entirely on human eyesight.

Consequently, the existing system fails to reflect today's advances in automation, particularly the progress in applying Artificial Intelligence models in agriculture, notably computer vision techniques for detecting irregularities and damage in agri-food products. Numerous investigations have examined the detection of diseases, defects, and damages in various agricultural items such as grains [2], crop leaves [3], fruits [4], and vegetables [5] by employing single-stage object detection models from the You Only Look Once (YOLO) [6] series. For example, in research focused on identifying Fusarium head blight in corn grains, Thangaraj Sundaramurthy *et al.* [7] applied the YOLOv5s algorithm, achieving an mAP@0.5 of 99%. Likewise, Sun *et al.* [8] employed the YOLOv5 model to recognize three categories of fungal damage in rice kernels, achieving mAP@0.5 values of 89.26, 91.15, and 90.19%, respectively. These investigations collectively demonstrate that YOLO models are effective tools for detecting damage and abnormalities in agricultural commodities. However, unmodified YOLO models prove impractical for use in limited-resource portable devices such as the Intel Neural Computing Stick, NVIDIA Jetson Nano, or Raspberry Pi, owing to their large model size and high Floating-Point Operations per Second (FLOPs), commonly known as computational cost. Hence, the readily available object detection models featured in those studies, while strong foundations for further development, cannot be effectively utilized in low-power and resource-limited hardware settings. This highlights the need to develop more efficient models with reduced cost and size by adjusting the network designs of standard object detection systems.

"Lightweight" models are created through three main network compaction approaches: 1. Model Compression, 2. Knowledge Distillation, and 3. Modification of Network Structures [9]. Over recent years, several agricultural studies have applied one of these strategies. For instance, to reduce the size and cost of the YOLOv7 model used for canola kernel damage detection, Thakuria and Erkinbaev [10] introduced the YOLOv7_Ours model by replacing convolutional layers in the ELAN and SPPCSPC modules with ghost layers, achieving a 32.1% decrease in model size and a 37.1% reduction in model cost. In research aimed at building a tea leaf detection model for Raspberry Pi 4 deployment, Zhang *et al.* [11] reduced the baseline YOLOv5 model size from 334 MB to 90.1 MB by redesigning the backbone with a ShuffleNetV2 feature extractor and applying channel pruning. Similarly, Xu *et al.* [12] incorporated the ShuffleNetV2 model into the YOLOv5s backbone for sugarcane aphid detection, yielding a 92.5% size reduction and an 81.6% cost reduction. Qi *et al.* [13] replaced the YOLOv5 backbone with GhostNet for agricultural pest detection, achieving 42% cost reduction and 39% size reduction.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Additionally, Jia *et al.* [14] used MobileNetV3 as a backbone replacement for the YOLOv7 model in rice pest detection. These works collectively show that redesigning network architectures with efficient, lightweight CNNs such as MobileNetV3, ShuffleNetV2, and GhostNet effectively reduces model size and cost, enabling deployment on embedded systems. Despite existing efforts to compact neural networks for mobile use, limited research fully demonstrates the end-to-end implementation of such optimized models.

Thus, the primary goal of this study was to develop a full end-to-end computer vision system for detecting canola seed damage. This objective includes three components: (i) Creation of two new datasets (D1 and D2) featuring damaged canola seeds prepared via two different crushing techniques (manual and semi-automatic), (ii) Training a YOLOv5s model to identify the two damage types, then enhancing the neural network structure for compatibility with resource-limited and portable devices by swapping the backbone — the main feature extraction component — with streamlined CNN architectures optimized for mobile or low-power environments, (iii) Integrating the resulting models onto an affordable GPU-powered edge device (NVIDIA Jetson Nano) to enable intelligent real-time canola quality assessment. The secondary goal was to compare the two sample preparation techniques (manual versus semi-automatic crushing). This included measuring the time needed for seed crushing and examining how each method influenced the accuracy of canola damage detection.

Materials and Methods

Sample preparation: handheld roller and semi-automatic machine

Initially, a collection of 20 paper strips, each containing 500 crushed canola kernels, was prepared using the standard handheld roller method. This roller was acquired from Dimo's Labtronics (Winnipeg, Manitoba, Canada) and features a platform containing 500 holes designed for positioning canola seeds before crushing. The procedure involves placing a small quantity of seeds on the perforated platform, covering them with masking tape (adhesive side down), and then rolling the setup firmly to crush the seeds. Each strip held 500 crushed canola seeds encompassing all three seed categories. Across the full set, a trained individual identified 380 immature seeds (3.80% of the total 10,000 crushed seeds) and 143 heated seeds (1.43%).

Next, another set of 20 identical strips was produced using a dedicated canola-crushing machine. The semi-automatic canola crusher (Devloo Canola Crusher, Manitoba, Canada) is constructed from aluminum and stainless steel. It consists of four main components — frame, seed hopper, crush mechanism, and pickup wheel. Operating the machine requires four crank turns to process 500 canola kernels. The crushing sequence proceeds as follows: during the initial crank rotation, seeds from the hopper move to the rotating pickup wheel. The tape fixed to the upper section serves as a carrier, guiding the seeds through the crushing rollers during subsequent crank turns. The seeds adhered to the tape were subsequently examined visually by an expert for any damage. In this batch, the trained person detected 337 immature seeds (3.37%) and 219 heated seeds (2.19%).

Image acquisition, annotation, and augmentation to create the datasets

Before training the detection model, the first requirement is to build a dataset that covers all three categories: sound kernels, heated kernels, and immature kernels. In contrast to the usual method of displaying individual classes across separate images for training and detection, the current work placed all three categories in a single shared image. Even though this choice increased the difficulty of model training, it was chosen to remain consistent with the established sample preparation rules for canola grading (Section 2.1).

Images of every strip were captured with a standard camera (Sony CMOS, 48 MP, f/1.78, with OIS) under uniform lighting conditions. Each strip sat directly beneath a 20 W broadband overhead light at a steady 30 cm distance. The collection from the initial sample creation process was labeled D1, and the one from the alternative preparation process was labeled D2. Each dataset ultimately contained 240 images, achieved by dividing each strip into three segments and

© 2024 The Author(s). | Published by Agro Intelligence Research Press

photographing them under four distinct background settings, as described in the creation process. Annotation of all target objects in the pictures relied on the Makesense AI tool to produce accurate ground-truth bounding boxes. An expert grader verified the entire annotation process against Canadian Grain Commission standards. The work was deliberately scheduled over six days, with two-hour daily sessions, to limit fatigue and reduce labeling errors. Short five-minute pauses followed every thirty minutes of effort. Special effort was taken to ensure that no erroneous annotations entered the ground truth data used for training, as faulty labels could seriously impair training effectiveness (Section 3.4). Moreover, the training dataset size was increased by applying augmentation operations, including horizontal and vertical flipping, image scaling, image shearing, image rotation, and image mosaicing, to all images across both datasets. The open-source Albumentation library handled these augmentation steps. Separate datasets (D1 and D2) were created specifically to examine the influence of the sample creation technique on model results and to support a side-by-side evaluation of the two techniques. Models trained on D2 are expected to surpass those trained on D1 thanks to superior dataset quality and the separated positioning of crushed canola kernels. Visual differences between the datasets are shown in **Figure 1**. An 8:2 split was used to create the training and validation sets for each dataset, which were then handled independently. Before augmentation, the total instances numbered 1079, including 717 immature class instances and 362 heated class instances. The augmentation step increased each instance count by 40%.

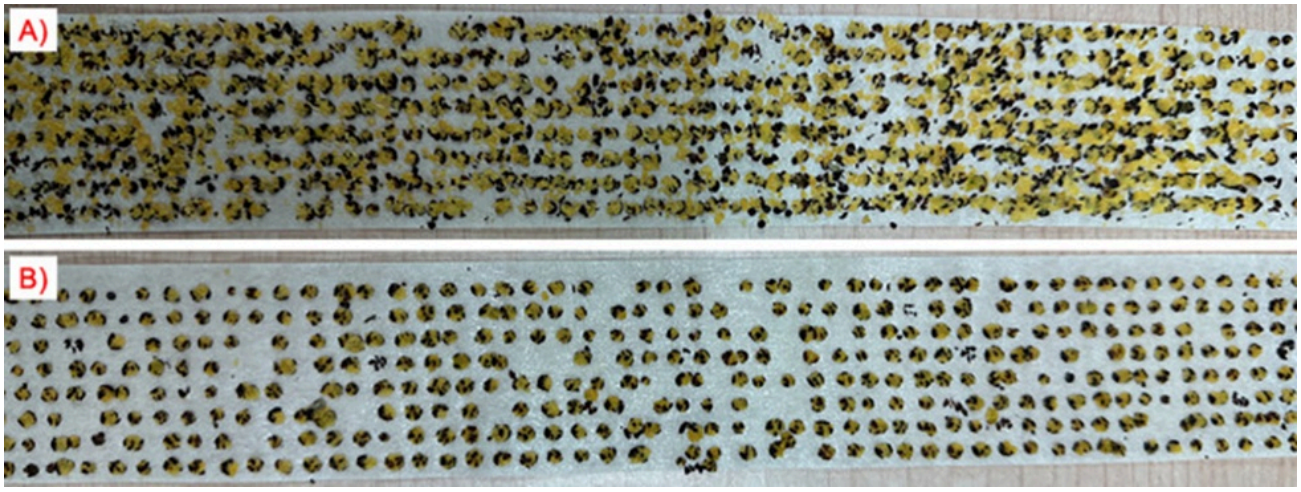


Figure 1. The two imaging datasets created from the two methods of sample preparation: (A) traditional handheld roller, and (B) semi-automatic canola crushing machine.

Sample preparation method comparison

Evaluation of the two preparation techniques used for samples and dataset building in this research (D1 and D2) focused on two aspects: (1) how well an object detection model identifies the desired objects and (2) the typical duration needed for sample creation. In the first part of the evaluation, separate object detection models underwent training on each dataset. In the second part, a stopwatch recorded the time it took an expert to generate 15 sample strips for each technique. Section 3.1 presents the more effective sample preparation approach.

Selection of object detection model for damage detection: one-stage vs two-stage networks

Two primary algorithmic strategies for object detection are one-stage and two-stage networks. A two-stage network first identifies multiple promising regions of interest that may hold the target object. Each region then passes through a convolutional neural network to determine whether its pixel content aligns with the ground-truth bounding box. Confirmed True Positive detections get assigned a class label and a bounding box that marks the object location within the image. Detection counts as correct once the intersection over union (IoU) score passes a defined threshold, generally 0.5 for tasks like this one. Popular two-stage detectors are based on the Region-based Convolutional Neural Network (R-CNN)

© 2024 The Author(s). | Published by Agro Intelligence Research Press

family. R-CNN relies on the Selective Search algorithm to generate roughly 2000 regions of interest from the supplied image before extracting features via a CNN from each one. A Support Vector Machine classifier then processes the CNN results to categorize the object, while a Linear Regression model adjusts the bounding box tightness. The multi-step process leads to an average detection time of 47 s per image for R-CNN on the COCO dataset [15]. Fast R-CNN emerged to fix the speed problem. In Fast R-CNN [16], the CNN extracts features before region proposals, after which a Softmax layer replaces the support vector classifier from the earlier version. Faster R-CNN [17] came next and uses a region proposal network to generate regions, while keeping the second stage identical to Fast R-CNN. Two-stage networks generally deliver higher accuracy but fall short of the rapid inference speeds required for real-time use, especially on edge computing hardware.

As a one-stage detector, the YOLO model family addresses the slow inference issue by treating object detection as a regression problem. Here, a single convolutional neural network determines bounding boxes and class probabilities for each target object during a single forward pass. The system achieves this by partitioning each image into an N x N grid, assigning detection responsibility to the cell containing the object's center. Output bounding boxes specify the object center position relative to the grid size, together with the object width and height scaled to the complete image. Non-maximal suppression prevents duplicate detections from any single grid cell [6]. Framing detection as single-stage regression enabled the original YOLO to achieve real-time performance of 45 frames per second (FPS), far exceeding the 0.5 FPS of the prior best Fast R-CNN model on the COCO dataset [18]. Although early YOLO delivered fast processing, it struggled with small groups of objects and could support only one object per grid cell. YOLO9000 [19] addressed these limitations by adding batch normalization layers, yielding a 2% mAP gain, and by enabling anchor boxes to support up to five objects per cell. Feature extraction in this version was based on the DarkNet-19 convolutional neural network, serving as the backbone. The backbone component of the YOLO model is a convolutional neural network that extracts features from the input image. YOLOv3 advanced feature extraction further by deepening the backbone to 106 layers via the Darknet53 architecture [20]. YOLOv4 [21] incorporated a neck section with a Path Aggregation Network (PAN) and Spatial Pyramid Pooling (SPP), positioned between the backbone and the head to merge feature maps. It additionally introduced Bag of Freebies (BoF) and Bag of Specials (BoS) concepts. BoF applies methods that improve mean Average Precision (mAP) with no extra training cost, such as mosaic data augmentation, the CIoU loss function, and self-adversarial training within the head. BoS, however, focuses on major performance gains with only modest speed trade-offs, primarily through Cross-stage partial (CSP) links in the Darknet53 backbone and Mish activation functions across both the backbone and the detector. Overall, these modifications delivered a 10% increase in mAP over YOLOv3. Released soon after YOLOv4, YOLOv5 marked the first shift to the PyTorch framework from the prior Darknet base. A key upgrade involved AutoAnchor, a new procedure for producing anchor boxes that better align with target object aspect ratios. AutoAnchor begins by applying KMeans clustering to ground truth bounding boxes and treats the cluster centers as seeds for a Genetic algorithm that refines and confirms the anchors. Another upgrade from YOLOv4 replaced SPP with SPPF (Spatial Pyramid Pooling-Fast). SPP and SPPF yield equivalent mathematical outcomes, yet SPPF demands fewer computational resources (in terms of FLOPS) by performing convolutions in sequence before merging results rather than handling them simultaneously.

YOLOv5s: the baseline canola damage detection model

The YOLOv5s (version 6.1) object detection model served as the foundation for detecting damaged kernels and was later adapted into a more compact form for embedded device use. This model is one member of the YOLOv5 series, which also offers nano, small, medium, large, and X-large options. Developed by Jocher *et al.* [22], it functions as a one-stage detector. Its structure follows earlier YOLO designs with three core sections: backbone, neck, and head. The backbone processes incoming images to extract layered features essential for object identification. YOLOv5s relies on the Cross-Stage Partial Darknet 53 (CSPDarknet53) [23] convolutional neural network as its backbone. The 53-layer deep CNN stacks its layers to establish a progressive feature hierarchy, with bottom layers focusing on basic details and upper layers managing higher-level abstract information. The neck performs multi-scale feature merging to strengthen understanding of image context, using Spatial Pyramid Pooling Fast (SPPF) [24] and the Cross-Stage Partial-Path Aggregation Network (CSP-PAN) [25]. SPPF generates pooled features across multiple scales, and PAN [26] integrates

© 2024 The Author(s). | Published by Agro Intelligence Research Press

spatial and channel details from multiple feature maps to improve overall detection quality. Finally, the model uses YOLOv3's detection head [20] to output bounding boxes, class probabilities, and objectness scores at multiple scales.

Improved ShuffleNetV2_YOLOv5s network architecture

The chosen backbone architecture plays a major role in determining the total model size and processing requirements. Naturally, "lightweight" CNN designs like ShuffleNet [27] and MobileNet [28] have fewer parameters and require less computation during both training and inference, potentially leading to a more compact model and lower computational demands than CSPDarknet53. These advantages prove especially useful for implementation on devices with limited resources, where storage, memory, and processing capabilities are restricted. Accordingly, this research rebuilt the original model's backbone with the ShuffleNetV2 network. ShuffleNetV2 [29] was picked because it offers an effective and compact CNN framework tailored for mobile and embedded systems (Section 3.2). ShuffleNetV1 [27], the prior iteration, was avoided here because it showed lower efficiency due to elevated memory access cost (MAC) and limited computational parallelism. These shortcomings arose from design elements, including pointwise (1 × 1) group convolutions, element-wise operations such as addition, and inconsistent channel widths due to bottleneck configurations. To address these drawbacks, ShuffleNetV2 introduced targeted modifications that improve parallelism and reduce memory access costs through smarter architectural decisions, notably a channel-splitting step after every unit. This divides feature channels into two paths: c-c' and c' (where c' equals c/2). One path stays as Identity while the other applies three convolutions, maintaining identical input and output channel counts to keep MAC low. The branches are then joined and subjected to a channel shuffle **(Figure 2a)**. The spatial down-sampling configuration appears in **Figure 2b**. Stacking these two building blocks creates the complete ShuffleNetV2 CNN. Integrating the ShuffleNetV2 backbone into the YOLOv5s network enhances parallelism during training and reduces MAC usage — a vital factor in object detection model training — resulting in significantly quicker, more efficient training. The ShuffleNetV2_YOLOv5s model comprises 16 ShuffleNetV2 units. Within the backbone area, the input image first passes through a module with 3 × 3 convolutions (kernel size = 6, stride = 2, padding = 2), batch normalization, a max-pooling layer, and a SiLU activation. The output tensor then enters four stacked ShuffleNetV2 units, followed by eight more and then four additional stacked units. The initial two units substitute the standard C3 modules from the original network and include a shortcut link to the neck section to avoid losing important features. The SPPF module remains unchanged since it operates faster than the original SPP and effectively combines features from all ShuffleNetV2 units. Details of the rebuilt YOLOv5s backbone based on the efficient ShuffleNetV2 CNN design, along with a side-by-side comparison, are presented in **Figure 3**, while the complete ShuffleNetV2_YOLOv5s model appears in **Figure 4**.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

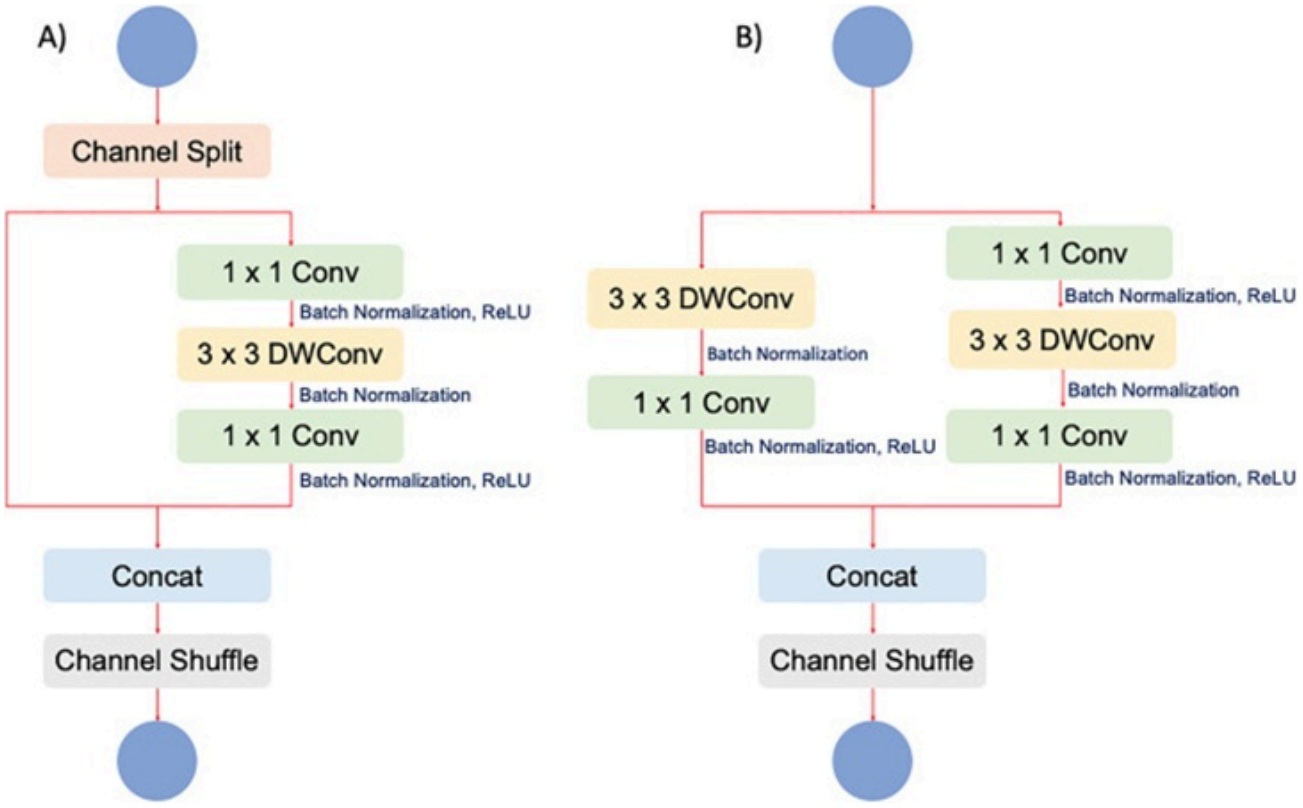


Figure 2. Architecture of ShuffleNetV2.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

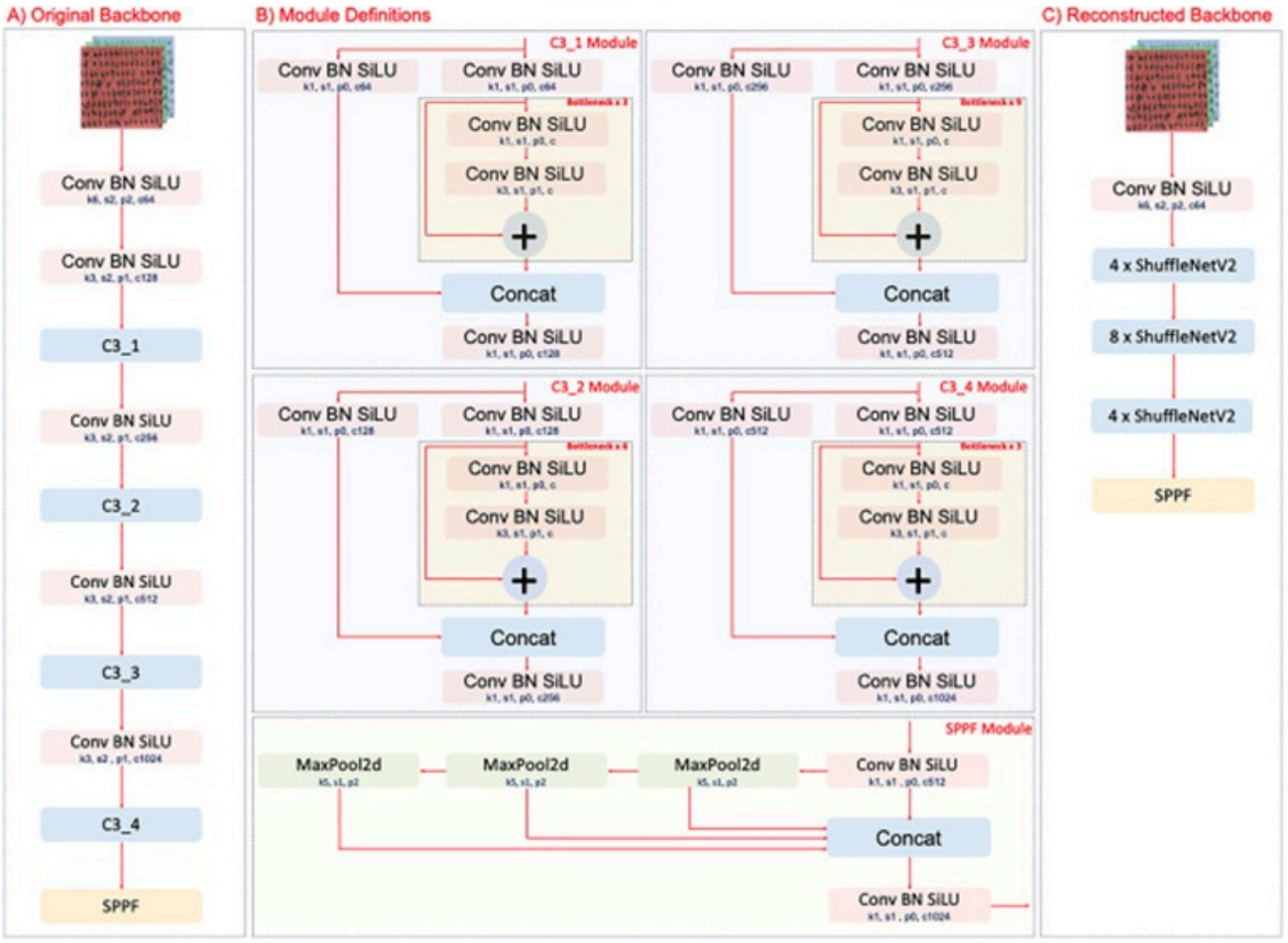


Figure 3. The comparison of the original and the reconstructed backbone architectures.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

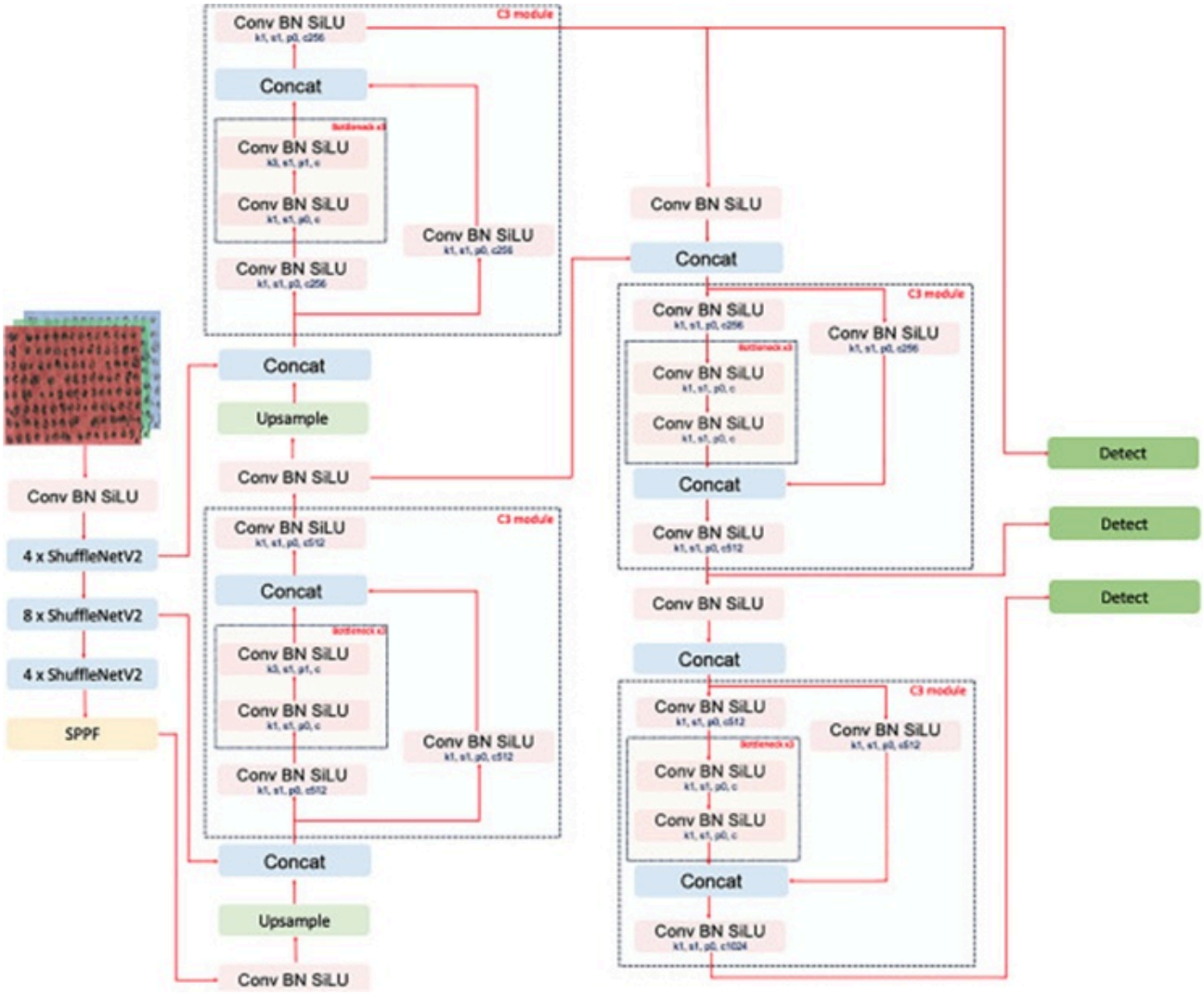


Figure 4. The structure of the ShuffleNetV2_YOLOv5s model.

Model training and performance evaluation

All models created in this study underwent training with identical hyperparameter settings (Epochs: 100, Batch size: 8, Image Size: 640×640 px, Initial and Final learning rate: 0.01, Optimizer: Stochastic Gradient Descent, and Momentum: 0.937) on an NVIDIA Tesla T4 GPU. Model performance assessment relied on several key indicators: Precision **(Eq. 1)**, Recall **(Eq. 2)**, F1 Score **(Eq. 3)**, mean Average Precision at 0.5 Intersection over Union (mAP@0.5IoU) **(Eq. 4)**, and Inference speed (pre-process time + inference time + non-maximal suppression time) **(Eq. 5)**.

$$Precision\ (P) = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (1)$$

$$Recall\ (R) = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (2)$$

© 2024 The Author(s). | Published by Agro Intelligence Research Press

$$F1\ Score = \frac{2*Precision*Recall}{Precision+Recall} \quad (3)$$

$$mAP = \frac{1}{2}\sum_{m=0}^{1}\int_{0}^{1} P_m\left(R_m\right) dR_m \quad (4)$$

$$FPS = \frac{1}{mDT} = \frac{1}{\frac{1}{M}\sum_{i=1}^{M} t_i} \quad (5)$$

Deployment in an edge-AI device and integration with the canola crusher

The Jetson Nano (Jetpack 4.6.1, Ubuntu 18.02, Python 3.8.12, PyTorch v0.9.0, PyTorch Vision v0.11.1), an edge-computing platform from NVIDIA Corporation, served as the deployment environment for the lightweight model. Operating at just 5 W, this device features a Quad-core ARM A57 CPU running at 1.46 GHz and a 128-core Maxwell GPU. After completing model training, the weight file from the top-performing lightweight model was converted from the .pt (PyTorch) format to the .onnx (Open Neural Network Exchange) format, then moved to the NVIDIA Jetson Nano. The conversion to .onnx was intended to improve the network's inference speed. This deployment step aimed to showcase the model's real-time capability to identify damage in canola samples precisely. The embedded system was smoothly integrated into the canola crusher equipment. This setup supported the following process: Canola kernels entered the machine through the hopper, after which the crank turned to crush them. The camera position allowed direct imaging of the moving strip containing crushed kernels. Such synchronized capture supported live oversight of the canola damage detection workflow, displaying detection boxes instantly on a monitor for quick review and assessment.

Results and Discussion

Comparison between methods of sample preparation and their effect on model performance

The traditional handheld roller and the semi-automatic canola crushing unit were evaluated as sample preparation techniques based on sample quality and preparation time. Results showed that the semi-automatic unit produced seeds that were more evenly crushed and properly aligned. At the same time, handheld roller samples often had seeds touching one another, with some remaining partially uncrushed. **Figure 5** illustrates the time required to create 15 sample strips, clearly demonstrating the machine's advantage by enabling higher sample throughput in less time and reducing operator fatigue. Additionally, faster processing enabled the generation of larger sample volumes for testing, providing a more accurate picture of real-world canola damage levels.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

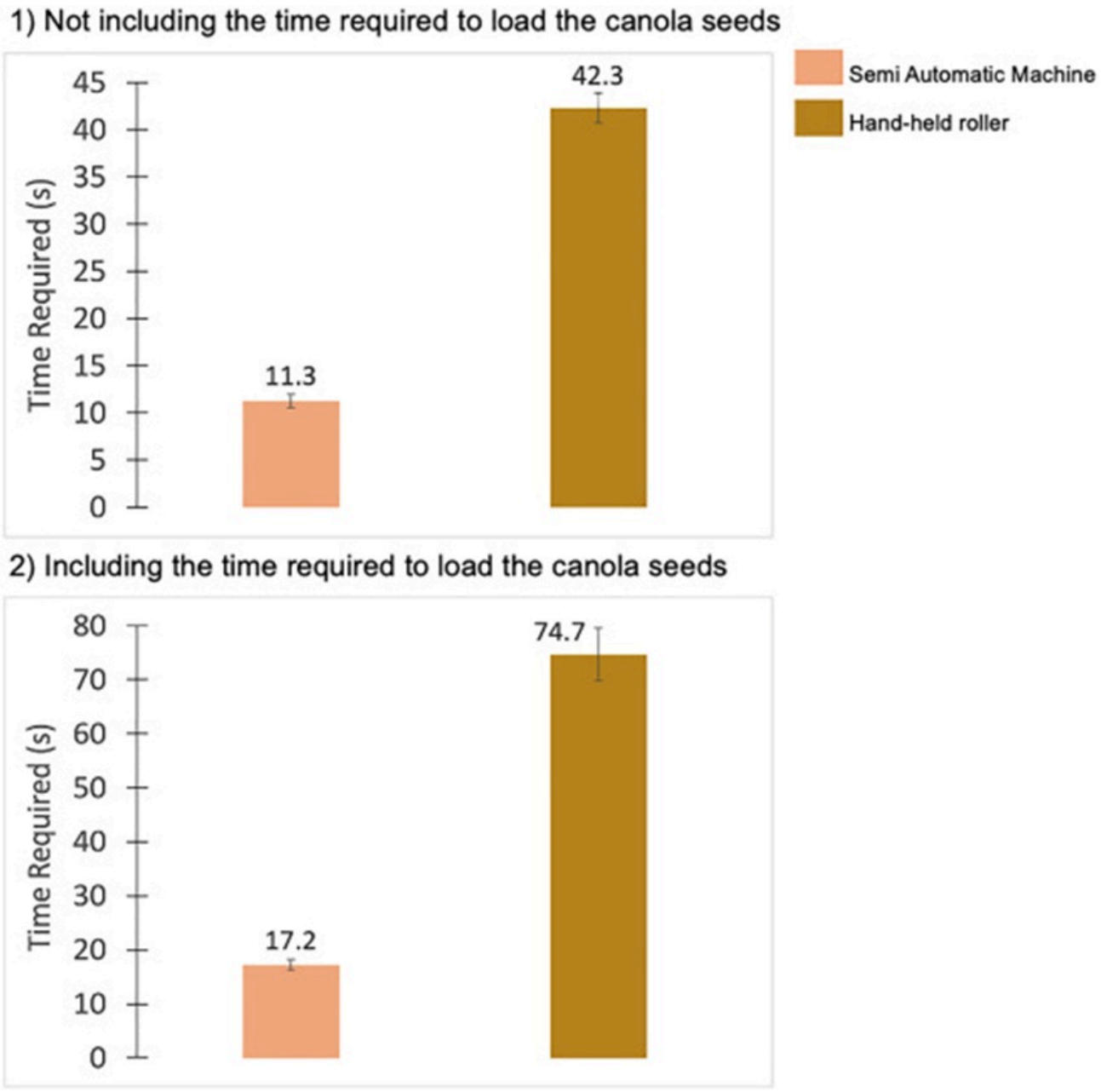


Figure 5. Comparison between the time required to prepare crushed canola samples using a handheld roller and a semi-automatic crushing machine, (A) not including the time required to load the canola seeds, and (B) including the time required to load the canola seeds.

Moreover, the base YOLOv5s model was trained separately on each dataset. Training behavior across both datasets is shown in **Figure 6**. Loss and mAP trends indicate that the model using D2 converged faster and exhibited lower fluctuations. **Figure 7** summarizes the YOLOv5s model outcomes on both datasets. The F1 score and mAP@0.5 values for the YOLOv5s model trained on D1 fell by 1.7% and 1.4%, respectively, below those from D2. This improvement stems from superior dataset quality in D2, which allows the model to better capture class-specific features through improved kernel alignment, consistent kernel dimensions, and the absence of kernel overlap. These findings align with earlier work by Thangaraj Sundaramurthy *et al.* [7], which reported stronger YOLOv5 performance on non-touching corn kernels compared to touching ones. Handheld roller preparation left some kernels uncrushed, leading the baseline model to mistakenly classify them as heated due to visual similarity between heated seed cotyledon color and dark brown shells.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Touching kernels further complicated annotation and increased the risk of mislabeling. Ground-truth boxes in D2 images were clearer, whereas D1 instances showed overlaps among classes, sound samples, and seed coats. Annotation challenges contributed to weaker metrics. In summary, performance differences arise from multiple factors in computer vision and deep learning. D2's well-defined, clearly visible target objects supplied richer, higher-quality training material, helping the YOLOv5s model learn distinct features more successfully and improve recognition accuracy. Conversely, D1's less distinct objects — caused by irregular crushed seed diameters — created training obstacles that limited effective feature extraction and prediction reliability. Since the baseline model showed stronger results on D2, subsequent lightweight CNN backbone experiments used this dataset. These combined factors enabled better detection of damaged kernels in the D2-trained model, as demonstrated in **Figure 8**. The D1-trained model produced 4 false negatives in a sample test image: 1 immature kernel and 3 heated kernels. All errors occurred with overlapping seeds, highlighting annotation difficulties and underscoring the need for accurate data labeling in computer vision tasks.

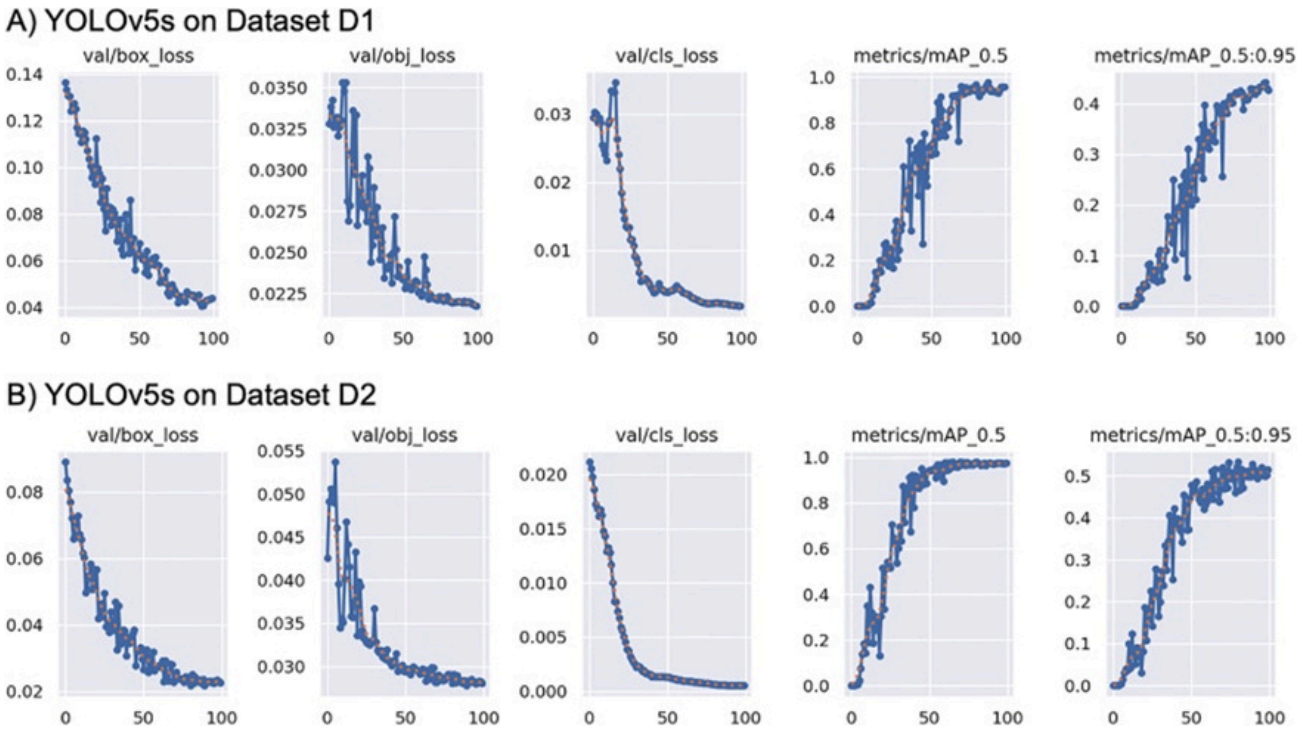
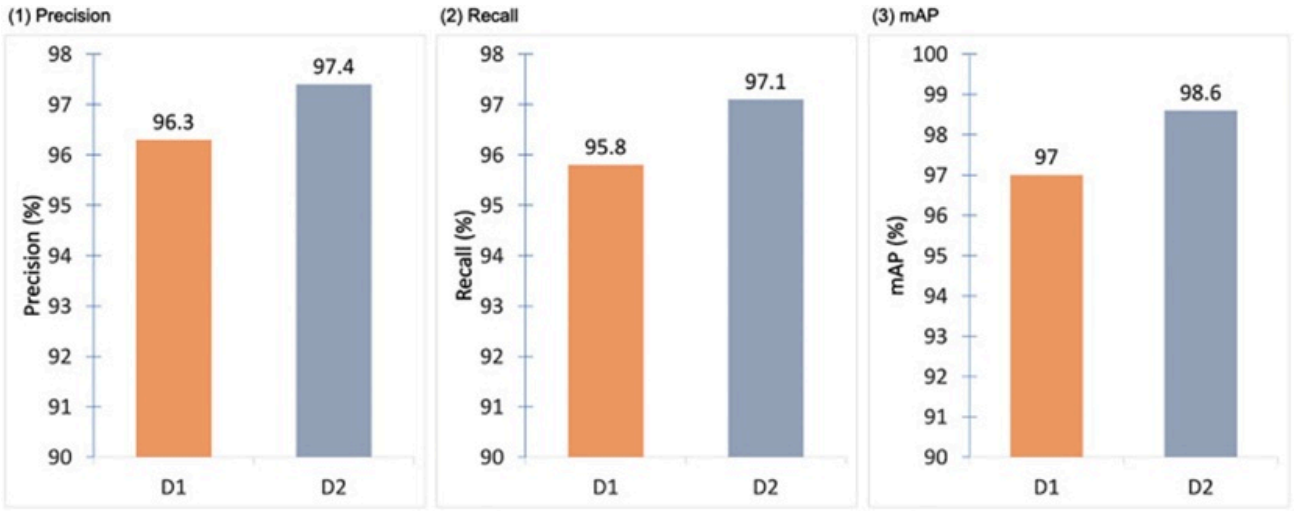


Figure 6. Difference in training metrics (Loss and mAP) for the YOLOv5s model trained on (A) dataset D1 and (B) dataset D2.



© 2024 The Author(s). | Published by Agro Intelligence Research Press

Figure 7. Performance of the base YOLOv5 model when trained on D1 and D2 Datasets.

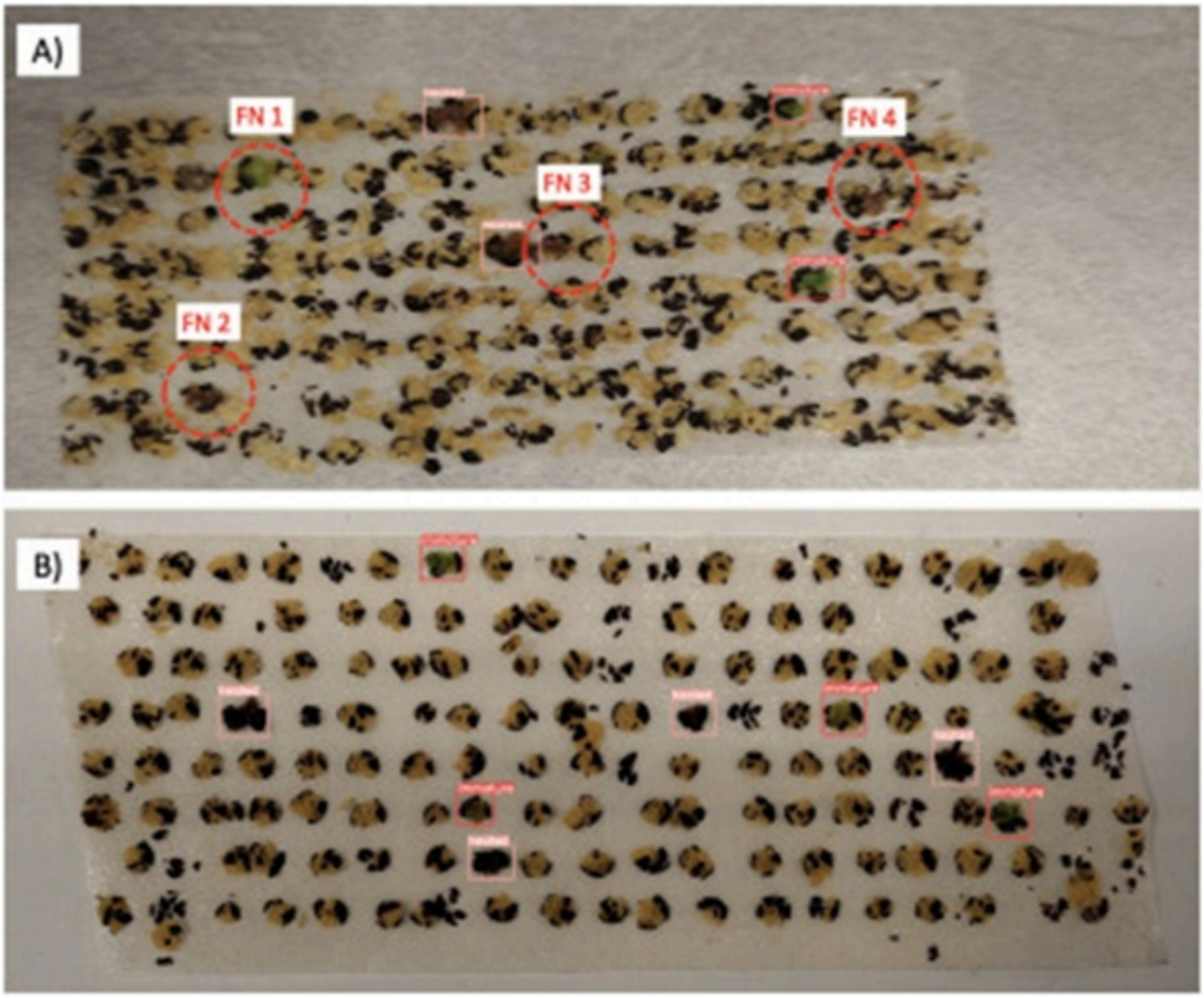


Figure 8. Detections made by the YOLOv5s model on a representative image present in A) test dataset D1 and B) test dataset D2. FN stands for False Negative.

Comparison between the baseline model and the lightweight model

Figure 9 presents the performance variations across the three models trained using the D2 dataset. The ShuffleNetV2_YOLOv5s model was identified as the optimal lightweight model because its size was reduced by 56.5% compared to the MobileNetV3_YOLOv5s model and by 80.4% compared to the baseline YOLOv5s model. In terms of computational complexity, the YOLOv5s model with the ShuffleNetV2 backbone achieved a 64.1% reduction compared to the YOLOv5s model with the MobileNetV3 backbone and a 99.5% reduction compared to the original model. Additionally, this model contained only 2,906,091 trainable parameters, corresponding to 56.7% and 82.95% fewer parameters than those in the MobileNetV3_YOLOv5s and YOLOv5s models, respectively. Regarding inference speed, the base YOLOv5s, ShuffleNetV2_YOLOv5s, and MobileNetV2_YOLOv5s models achieved 56.1, 98, and 53.7 FPS, respectively, on a validation dataset using an NVIDIA Tesla T4 GPU.

Nevertheless, owing to the four-way trade-off among model size, model cost, inference speed, and overall accuracy, the selected lightweight model could not exceed the other two in mAP@0.5, registering values 0.9% lower than the baseline

© 2024 The Author(s). | Published by Agro Intelligence Research Press

model and 0.5% lower than the alternative lightweight model. In F1 score, it ranked second-best, behind only the original model, with a result 0.4% below YOLOv5s yet 0.9% above the other lightweight alternative. Still, this modest decline in both mAP and F1 scores relative to the original network may be overlooked in view of the major cuts in size and cost metrics, which enabled effective deployment on resource-limited embedded devices. Furthermore, the 97.7% mAP delivered by the ShuffleNetV2_YOLOv5s model surpasses the accuracy expected from an expert grader working in both fatigued and non-fatigued states. Visualizations of the detections generated by all three models on an identical representative test image are shown in **Figure 10**. The single false negative case, observed only with the MobileNetV2_YOLOv5s model weights, occurred because that model achieved the lowest F1 score. Consequently, among the two lightweight options, the ShuffleNetV2_YOLOv5s model stood out as the stronger choice, surpassing the MobileNetV3_YOLOv5s model across all metrics except mAP.

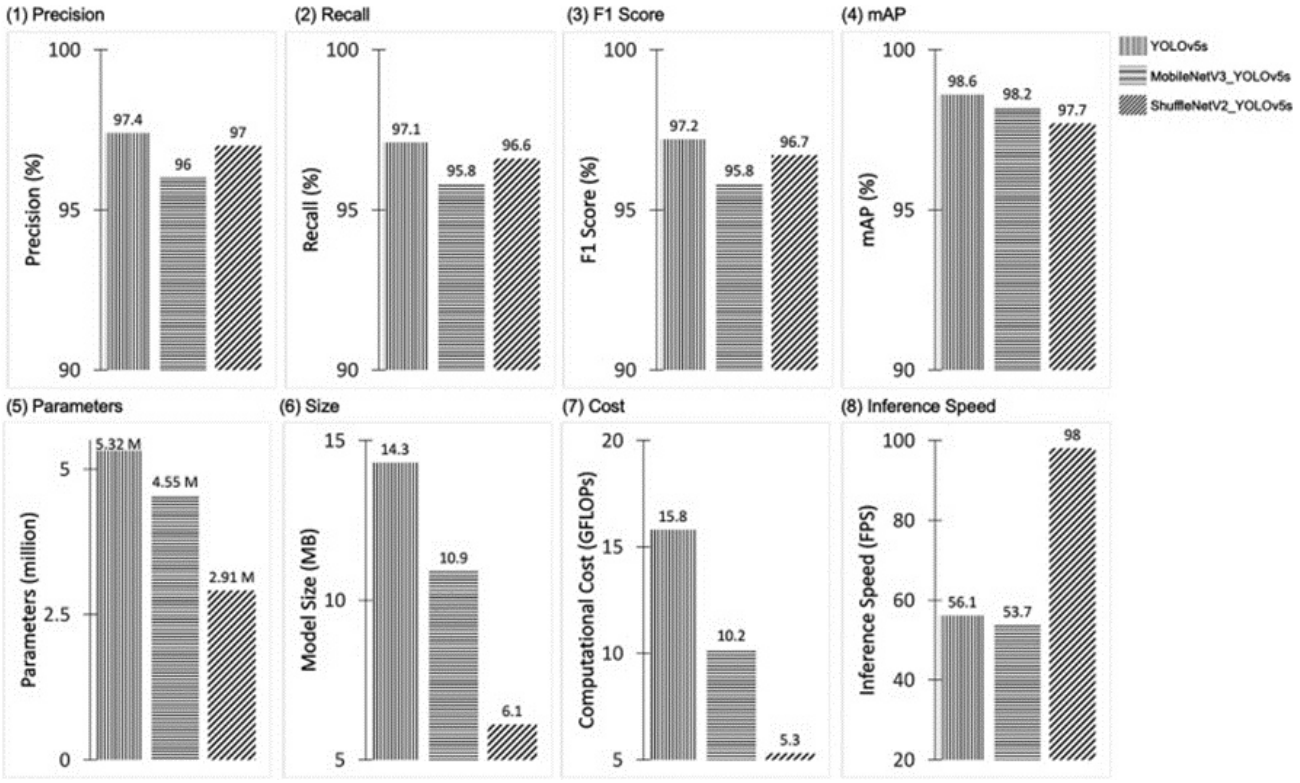


Figure 9. Comparison of lightweight models (MobileNetV3-YOLOv5s and ShuffleNetV2_YOLOv5s) with the Base YOLOv5s model.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

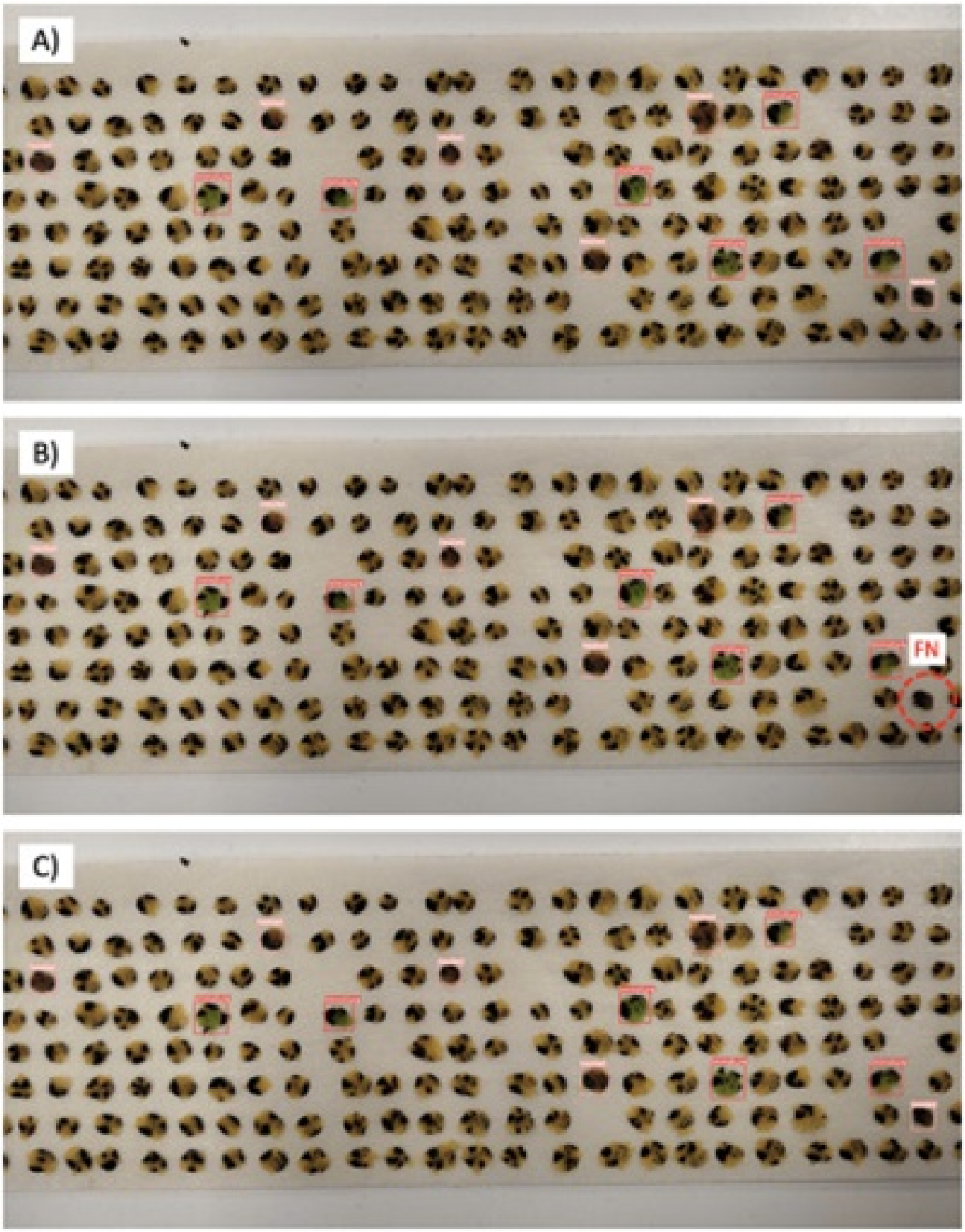


Figure 10. Detections made by (A) Base YOLOv5s, (B) MobileNetV3_YOLOv5s, and (C) ShuffleNetV2_YOLOv5s on the same representative test image.

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Deployment results, hardware limitations, and benefits of inferencing on the edge

The complete system functioned successfully during operation, displaying neither false positives nor false negatives across the real-time test videos acquired by the onboard camera. This result contrasted with the model's validation dataset performance and was associated with increased inference time on the edge device, compared with the Tesla T4 GPU's 98 FPS. As a result, the elevated detection speed on the Maxwell GPU produced superior accuracy. The baseline YOLOv5s model with the CSPDarknet53 backbone achieved only 1.25 frames per second, the lowest among the three models, versus 5 frames per second for the ShuffleNetV2_YOLOv5s model. This fourfold improvement in FPS substantiates the effectiveness of the network compaction applied to the baseline. The ShuffleNetV2_YOLOv5s model's performance on the Jetson Nano was further evaluated against the same model running on a Raspberry Pi 4. Inference speed on the Jetson Nano exceeded that on the Raspberry Pi 4 by 133.3%, with the former processing damaged seeds at 5 FPS, while the latter managed only 1 FPS. This gap arises from fundamental hardware differences between the platforms. The Jetson Nano includes a dedicated Maxwell GPU with 128 CUDA cores optimized for parallel processing typical in neural network operations. Such parallelism greatly speeds up matrix multiplication, convolution, and element-wise operations, which are central to convolutional networks. Moreover, the Jetson Nano integrates a quad-core ARM Cortex-A57 CPU that collaborates with the GPU on AI tasks. This CPU handles task offloading and efficient memory coordination, yielding better inference results than the Raspberry Pi 4, which offers less specialized AI hardware support and therefore slower processing. It should be emphasized, however, that inference speed is largely hardware-dependent—for example, the top lightweight model reached 98 FPS on the Tesla T4 GPU used for training, but only 5 FPS on the Jetson Nano's less powerful GPU. Higher FPS is possible with stronger edge hardware, such as the NVIDIA Jetson AGX Orin, but this would dramatically increase system costs and may prove unaffordable for farmers. For example, the Jetson AGX Orin costs 16 times as much as the device used in this research. Even so, the lightweight model from this study can be straightforwardly implemented on such high-end embedded platforms for near-real-time edge inferencing. Independent of how edge devices compare in detection performance to cloud or workstation GPUs, edge inferencing offers multiple advantages over cloud-based processing. A key advantage is substantially reduced latency, which suits real-time applications such as automated grading of agricultural goods. Edge computing also bolsters privacy and security by enabling local data retention, minimizing breach risks, and reducing bandwidth usage by transmitting only essential data, making it economical and practical in areas with unreliable internet access.

Effect of data quality and data annotation on model performance

A clear link exists among image quality, annotation quality, imaging conditions during training, and the resulting model performance when trained under matching conditions. Our research outcomes align with observations from multiple prior studies on how dataset quality factors influence model effectiveness. For example, research by Ma *et al.* [30] investigating the effects of data annotation revealed that reannotating 5000 images from the Google Open Images dataset markedly boosted the mean average precision score. The reannotation corrected numerous incorrect labels and ambiguities, thereby improving overall dataset quality. This method of enhancing training data—through model adjustments and improvements such as noise reduction or expanding the set of trainable examples—is called Data-centric AI and is gaining momentum as a pathway to affordable AI development. Likewise, Agnew *et al.* [31] reported that applying radial and uniform noise to annotations in the COCO2017 dataset harmed the mean average precision of the RCNN object detection model. Consistent with these findings, data annotation quality on D2 far exceeded that on D1, resulting in a 1.4% improvement in mAP for the same model. In addition, the effects of input image size were analyzed, with YOLOv5s model results summarized in **Table 1**. Across both datasets, the smallest input resolution (i.e., 320 pixels) yielded the weakest results, confirming that lower-quality image data during training leads to inferior F1 scores and mAP. Patterns revealed an empirical connection between model performance and input image resolution in both datasets, yet no evidence emerged that performance scales proportionally with image size.

Table 1. The Effect of Model Input Image Resolution on Model Performance for both datasets.

Dataset	Image Size	P (%)	R (%)	F1 Score (%)	mAP@0.5 (%)

© 2024 The Author(s). | Published by Agro Intelligence Research Press

D1	320×320	94.8	94.8	94.8	95.8
	512×512	95.5	95.2	95.3	96.4
	640×640	96.3	95.8	96.1	97.0
D2	320×320	95.0	94.8	94.9	96.4
	512×512	96.2	96.4	96.3	98.2
	640×640	97.4	97.1	97.2	98.6

Conclusion

Canola quality is evaluated based on the number of damaged (immature and heated) seeds. Multiple factors contribute to this, including frost at harvest time and inadequate storage conditions that lead to enzyme inactivation. Such damage affects market grade and price because the extracted oil requires additional processing, which lowers yield and oil quality while increasing costs. Accurate detection is therefore vital for proper grading, a process historically reliant on manual labor for sample preparation and seed damage assessment. This study introduced an end-to-end computer vision method for identifying damaged canola seeds. It featured a new semi-automatic canola crusher machine for sample preparation, which outperformed conventional techniques in both preparation time and sample quality. The damage identification automation was approached as an object-detection task, solved using the YOLOv5s model. Given the need for deployment in resource-limited settings with real-time capability, the base model underwent compaction to lower size and complexity. This involved replacing the backbone with the ShuffleNetV2 CNN architecture. The outcome was a compact, efficient model well-suited for embedded systems. This lightweight version accurately detected all damages in live video from the camera attached to the Jetson Nano device at an acceptable speed. Overall, the work lays the foundation for similar comprehensive frameworks to automate damage detection across various agricultural commodities.

Acknowledgements

None

Conflict of interest

None

Financial support

None

Ethics statement

None

Received: 08 Jul 2023 Revised: 12 Sep 2023 Accepted: 28 Sep 2023
Published online: 25 November 2024

© 2024 The Author(s). | Published by Agro Intelligence Research Press

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at https://creativecommons.org/licenses/by-nc-sa/4.0/.

References

1. Chung DW, Pružinská A, Hörtensteiner S, Ort DR. The role of pheophorbide a oxygenase expression and activity in the canola green seed problem. Plant Physiol. 142(1):88-97. https://doi.org/10.1104/pp.106.084483.

2. Liu Z, Wang S. Broken corn detection based on an adjusted YOLO with focal loss. IEEE Access. 2019;7:68281-9. https://doi.org/10.1109/ACCESS.2019.2916842.

3. Mathew MP, Mahesh TY. Leaf-based disease detection in bell pepper plant using YOLO v5. Signal Image Video Process. 2022;16(3):841-7. https://doi.org/10.1007/s11760-021-02024-y.

4. Hu WX, Xiong JT, Liang JH, Xie ZM, Liu ZY, Huang QY, et al. A method of citrus epidermis defects detection based on an improved YOLOv5. Biosyst Eng. 2023;227:19-35. https://doi.org/10.1016/J.BIOSYSTEMSENG.2023.01.018.

5. Li J, Qiao Y, Liu S, Zhang J, Yang Z, Wang M. An improved YOLOv5-based vegetable disease detection method. Comput Electron Agric. 2022;107345. https://doi.org/10.1016/J.COMPAG.2022.107345.

6. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. p. 779-88.

7. Thangaraj Sundaramurthy RP, Balasubramanian Y, Annamalai M. Real-time detection of Fusarium infection in moving corn grains using YOLOv5 object detection algorithm. J Food Process Eng. 2023;46(9). https://doi.org/10.1111/jfpe.14401.

8. Sun K, Zhang YJ, Tong SY, Tang MD, Wang CB. Study on rice grain mildewed region recognition based on microscopic computer vision and YOLO-v5 model. Foods. 2022;11(24):4031. https://doi.org/10.3390/foods11244031.

9. Zhang K, Ying H, Dai HN, Li L, Peng Y, Guo K, et al. Compacting deep neural networks for Internet of Things: Methods and applications. IEEE Internet Things J. 2021;8(15):11935-59.

10. Thakuria A, Erkinbaev C. Improving the network architecture of YOLOv7 to achieve real-time grading of canola based on kernel health. Smart Agric Technol. 2023;5:100300. https://doi.org/10.1016/j.atech.2023.100300.

11. Zhang S, Yang H, Yang C, Yuan W, Li X, Wang X, et al. Edge device detection of tea leaves with one bud and two leaves based on ShuffleNetv2-YOLOv5-Lite-E. Agronomy. 2023;13(2):577. https://doi.org/10.3390/agronomy13020577.

12. Xu W, Xu T, Thomasson JA, Chen W, Karthikeyan R, Tian G, et al. A lightweight SSV2-YOLO based model for detection of sugarcane aphids in unstructured natural environments. Comput Electron Agric. 2023;211:107961. https://doi.org/10.1016/j.compag.2023.107961.

13. Qi F, Wang Y, Tang Z, Chen S. Real-time and effective detection of agricultural pest using an improved YOLOv5 network. J Real Time Image Process. 2023;20(2). https://doi.org/10.1007/s11554-023-01264-0.

14. Jia L, Wang T, Chen Y, Zang Y, Li X, Shi H, et al. MobileNet-CA-YOLO: An improved YOLOv7 based on the MobileNetV3 and attention mechanism for rice pests and diseases detection. Agriculture. 2023;13(7):1285. https://doi.org/10.3390/agriculture13071285.

15. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014:580-7.

16. Girshick R. Fast R-CNN. Proceedings of the IEEE International Conference on Computer Vision. 2015. p. 1440-8.

17. Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst. 2015;28.

18. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft COCO: Common objects in context. In: Computer Vision-ECCV 2014: 13th European Conference, Zurich, Switzerland. Springer International Publishing. 2014; 13:740-55. https://doi.org/10.1007/978-3-319-10602-1_48.

19. Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. p. 7263-71.

20. Redmon J, Farhadi A. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. 2018.

21. Bochkovskiy A, Wang CY, Liao HYM. YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. 2020.

22. Jocher G, Chaurasia A, Stoken A, Borovec J, NanoCode012, Kwon Y, et al. Ultralytics/yolov5: v6.1 - TensorFlow Edge TPU and OpenVINO export and inference. 2022. https://doi.org/10.5281/ZENODO.6222936.

23. Wang CY, Liao HYM, Yeh IH, Wu YH, Chen PY, Hsieh JW. CSPNet: A new backbone that can enhance learning capability of CNN. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020:390-1.

24. He K, Zhang X, Ren S, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell. 2015;37(9):1904-16.

25. Wang CY, Bochkovskiy A, Liao HYM. Scaled-YOLOv4: Scaling cross stage partial network. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021:13029-38.

26. Liu S, Qi L, Qin H, Shi J, Jia J. Path aggregation network for instance segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. p. 8759-68.

27. Zhang X, Zhou X, Lin M, Sun J. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. p. 6848-56.

28. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861. 2017.

29. Ma N, Zhang X, Zheng HT, Sun J. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018. p. 116-31.

30. Ma J, Ushiku Y, Sagara M. The effect of improving annotation quality on object detection datasets: A preliminary study. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. 2022;19-

© 2024 The Author(s). | Published by Agro Intelligence Research Press

20:4850-9.

31. Agnew C, Eising C, Denny P, Scanlan A, Van De Ven P, Grua EM. Quantifying the effects of ground truth annotation quality on object detection and instance segmentation performance. IEEE Access. 2023;11:25174-88. https://doi.org/10.1109/ACCESS.2023.3256723.

© 2024 The Author(s). | Published by Agro Intelligence Research Press