

Article

CQ-RT-DETR: Lightweight Cotton-Field Weed Detection with Query Competition Calibration

Yangbin Li  and Linjing Wei * 

School of Information Science and Technology, Gansu Agricultural University, Lanzhou 730070, China

* Correspondence: wlj@gsau.edu.cn

Abstract

To address storage, computational, and latency constraints in cotton-field robots and variable-rate spraying, this study proposes CQ-RT-DETR with a two-stage design of structural compression and training-time query calibration. First, the hidden dimension of RT-DETRv2 is reduced from 256 to 192 and the number of object queries from 300 to 100, forming the lightweight B_Lite baseline. Next, the Scale-Reliability-Guided Trajectory-Consistent Competition Margin (SR-TCCM) mechanism selects persistent competing queries using cross-layer IoU trajectories and applies a budget-constrained adaptive margin based on object scale and the winner query's localization advantage. Across five random seeds on the test set, the mean AP_{50–95} values of RT-DETRv2, B_Lite, and CQ-RT-DETR are 0.8867, 0.8691, and 0.8892, respectively. CQ-RT-DETR improves AP_{50–95} by 2.01 percentage points over B_Lite and achieves accuracy comparable to the original RT-DETRv2, while reducing the parameter count and GFLOPs by 16.93% and 23.38%, respectively. In FP16 RKNN model-level forward-pass tests on an RK3576 platform, mean latency is 54.57% lower than for RT-DETRv2, and FPS increases from 6.69 to 14.7. These results show that CQ-RT-DETR recovers the compression-induced accuracy loss without increasing inference-graph complexity and provides a favorable accuracy–efficiency trade-off.

Keywords: cotton field weed detection; RT-DETRv2; lightweight object detection; query competition calibration; SR-TCCM; precision agriculture

1. Introduction

Cotton is an important fiber and oilseed crop, while weeds constitute one of the major biotic stresses affecting stable cotton production and efficient field management. During the seedling and vegetative growth stages, weeds compete with cotton plants for light, water, nutrients, and growing space, thereby reducing cotton yield and increasing the costs of manual and chemical weed control [1,2]. Long-term reliance on uniform whole-field herbicide application also increases chemical use. For precision weed management involving variable-rate spraying, mechanical weeding, and field robots, weed recognition systems must provide both class labels and spatial locations rather than merely perform binary weed/non-weed classification. Accordingly, multiclass weed detection in cotton fields is an essential sensing capability for operations tailored to weed type, location, and treatment demand [3–6].

Vision-based weed recognition has evolved from handcrafted to deep representations and from image classification to spatial localization. Early methods distinguished crops from weeds using color indices, texture, leaf shape, and vegetation-region segmentation, but were sensitive to variations in illumination, soil background, leaf overlap, and growth



Academic Editor: Jayme Garcia Arnal Barbedo

Received: 9 August 2026

Revised: 1 September 2026

Accepted: 4 September 2026

Published: 9 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

stage [5,7]; natural field images also commonly exhibit substantial variation in object scale and class imbalance [8]. The introduction of convolutional neural networks enabled models to learn plant phenotypic features directly. Dyrmann et al. used a deep convolutional network to classify seedlings from multiple crop and weed species [9], while Milioto et al. and Lottes et al. subsequently applied fully convolutional networks to real-time semantic segmentation of crops and weeds in field environments and demonstrated the feasibility of onboard robotic operation [10,11]. The public DeepWeeds dataset advanced research on multiclass weed recognition in natural backgrounds [12], while transfer learning alleviated training difficulties arising from limited agricultural datasets and variations in image-acquisition conditions [13]. Although these studies established the foundation for intelligent weed recognition, image classification cannot localize individual targets and semantic segmentation requires costly pixel-level annotations; object detection, which predicts both classes and bounding boxes, has therefore become an increasingly standard component of precision spraying and automated weeding equipment.

In cotton production, research has likewise shifted from binary crop–weed discrimination toward species-level weed recognition. Chen et al. established an image-classification benchmark comprising 15 common cotton-field weed species, demonstrating that deep transfer learning can resolve fine-grained interspecific differences and that class imbalance and model complexity substantially affect recognition performance [14]. Subsequently, Dang et al. developed CottonWeedDet12 and systematically evaluated 18 YOLO detectors on a common dataset; across the evaluated models, mAP@0.5 ranged from 88.14% to 95.22%, while mAP@0.5:0.95 ranged from 68.18% to 89.72% [15]. This benchmark demonstrates that multiple cotton-field weed species can already be detected with high accuracy, but it also exposes a persistent trade-off: high-capacity models generally provide stronger fine-grained representation and localization, whereas smaller models better suited to field devices tend to lose accuracy on small or occluded objects and underrepresented classes.

Recent cotton-field weed detection studies have predominantly pursued lightweight YOLO-based designs to balance accuracy against deployment cost. YOLO-WDNet reduces model size through a lightweight backbone, feature fusion, and attention mechanisms and was validated on a ground robotic platform [16]; YOLO-WL combines EfficientNet with multiscale attention to reduce the parameter count and accelerate inference [17]; and Cotton Weed-YOLO improves recognition efficiency by redesigning its feature-extraction and fusion structures [18]. By restructuring the YOLOv8 backbone and detection head, Star-YOLO demonstrated the potential to jointly achieve a lightweight design and high accuracy on CottonWeedDet12 [19]. More recently, AVGS-YOLO and YOLO-CottWed have continued along this line by incorporating lightweight convolutions, attention mechanisms, and efficient feature fusion [20,21]. These studies indicate that weed detection is no longer judged solely by whether weeds can be detected, but by whether detection remains reliable under limited computational resources; however, most approaches retain newly introduced convolutional, attention, or fusion modules in the inference graph, thereby coupling structural compression with accuracy compensation.

Comparative studies using real field imagery also indicate that YOLO is not the only viable option for real-time weed detection. Allmendinger et al. compared YOLOv8, YOLOv9, YOLOv10, and RT-DETR across 16 field plant species and found that the architectures offered distinct trade-offs in accuracy, recall, latency, and hardware suitability [22]. Zhang et al. further proposed CEAM-DETR and demonstrated the feasibility of a lightweight end-to-end Transformer for weed detection in soybean fields [23]. In parallel, Zhang et al. developed SCR-DETR from RT-DETR by reconstructing the backbone and encoder with SRC re-parameterized convolutions, AIFI-USAR attention, and CGSR multiscale fusion, reducing

model complexity on CottonWeedDet12 while maintaining mAP@0.5 and demonstrating real-time inference on a Jetson Nano [24].

Detection Transformer (DETR) formulates object detection as a set-prediction problem and uses bipartite matching to establish one-to-one assignments between predictions and ground-truth objects, thereby simplifying manually designed components such as anchors and non-maximum suppression [25]. Deformable DETR improves multiscale feature modeling through sparse sampling around reference points while enhancing small-object detection and training convergence [26]. RT-DETR subsequently introduced an efficient hybrid encoder and query-selection mechanism for real-time end-to-end detection [27]; RT-DETRv2 strengthened the baseline through improved training strategies and scale-adaptive configurations [28]; and RT-DETRv3 alleviated the sparse-supervision problem of one-to-one matching through hierarchical dense positive supervision [29]. Collectively, these studies demonstrate the potential of end-to-end detectors to balance accuracy, speed, and deployment simplicity. Building on this progress, the present study focuses on structural compression of RT-DETRv2 for close-range cotton-field imagery and on the resulting changes in query behavior.

Close-range cotton-field images generally contain only a few weed instances, whereas standard detectors retain far more queries than there are ground-truth objects. After the query budget is compressed, multiple queries may still compete around the same object, with one winner query ultimately receiving the matching supervision. If competition is insufficiently constrained, several candidate queries may retain similar scores and disrupt confidence ranking; conversely, uniformly strong suppression of all competing queries may allow a winner with an incidental early advantage to suppress candidates that still contain useful localization information. Among studies of matching, alignment, and ranking in DETR, Stable-DINO introduces localization indicators such as IoU into positive-sample classification supervision and matching costs to improve matching stability across decoder layers [30]; Align-DETR uses an Align Loss that jointly considers classification and localization quality, together with many-to-one supervision in intermediate layers, to mitigate classification-localization misalignment and target inconsistency across layers [31]; and Rank-DETR reinforces the ranking priority of predictions with high localization quality through the network architecture, loss function, and matching cost [32]. These methods address cross-layer matching stability, classification-localization alignment, and prediction ranking, respectively. However, they are not designed to analyze persistent competition around the same ground-truth object after query-budget compression, nor do they directly answer which queries remain in competition or how strongly that competition should be constrained. On this basis, the present study treats query competition in the compressed model as a mechanistic hypothesis to be tested experimentally. Without modifying Hungarian matching or the inference architecture, it uses cross-layer IoU trajectories to select persistent competing queries, defines an adaptive margin according to object scale and the winner query's localization advantage, and limits the calibration strength through a loss budget.

Based on the above analysis, this study proposes CQ-RT-DETR for multiclass cotton-field weed detection and adopts a two-stage strategy that first compresses the inference architecture and then compensates for the resulting accuracy loss through a training-time mechanism. In the first stage, the lightweight B_Lite baseline is constructed by reducing the hidden dimension of RT-DETRv2 from 256 to 192 and the number of object queries from 300 to 100. This design reduces the parameter count from 20.097 M to 16.694 M and GFLOPs from 22.947 to 17.581. In the second stage, the Scale-Reliability-Guided Trajectory-Consistent Competition Margin (SR-TCCM) mechanism analyzes cross-layer query IoU trajectories only during training and adjusts the ranking-constraint strength according to object scale and the winner query's localization advantage. With a fixed random seed of 42 on the

validation set, AP50–95 increases from 0.8684 for B_Lite to 0.8842. Across five random seeds on the test set, CQ-RT-DETR improves AP50–95 by an average of 2.012 percentage points over B_Lite, whereas its difference from RT-DETRv2 is not significant. In FP16 model-level tests on an RK3576 platform, mean latency decreases from 149.44 ms to 67.89 ms and FPS increases from 6.69 to 14.7.

The main contributions of this study are as follows:

- (1) We develop B_Lite, a lightweight RT-DETRv2 architecture for multiclass cotton-field weed detection. By jointly compressing the hidden dimension and query budget, B_Lite establishes a lightweight inference baseline with 16.93% fewer parameters and 23.38% fewer GFLOPs, while enabling quantitative analysis of the associated accuracy loss and scale sensitivity.
- (2) We propose SR-TCCM, a Scale-Reliability-Guided Trajectory-Consistent Competition Margin calibration mechanism. The method uses IoU trajectories across decoder layers to select persistent competing queries and adjusts the competition strength according to object-scale reliability and the winner query's localization advantage, thereby improving query ranking in the lightweight model without introducing additional inference modules.
- (3) We introduce a decoupled design that combines inference-architecture compression with training-time accuracy compensation. The SR-TCCM branch is completely removed during inference, allowing the final CQ-RT-DETR model to recover the compression-induced accuracy loss and maintain accuracy comparable to that of the original model while preserving the lightweight inference graph.
- (4) We conduct structural comparisons, scale analysis, five-seed independent testing, comparisons with mainstream detectors, and RK3576 model-level forward-pass tests on CottonWeedDet12, providing comprehensive evidence of the accuracy–efficiency trade-off in multiclass cotton-field weed detection.

2. Materials and Methods

2.1. Multiclass Cotton-Field Weed Detection Task and Dataset Partitioning

The task investigated in this study is to simultaneously classify and localize multiple weed species in RGB images acquired from natural cotton fields. Given a single field image as input, the model predicts the class confidence score and spatial location of each weed instance, providing essential information for variable-rate spraying, mechanical actuator positioning, and field-robot perception. Compared with binary weed/non-weed classification, multiclass detection must additionally address morphological similarities between classes, substantial within-class appearance variations across growth stages and object scales, and background interference jointly caused by soil, cotton plants, shadows, and occlusion.

To ensure experimental reproducibility, this study uses the publicly available CottonWeedDet12 dataset. The dataset was collected using smartphones or handheld digital cameras under natural field illumination in the southern United States between June and September 2021. The dataset contains 5648 RGB images, 12 cotton-field weed classes, and 9388 bounding-box annotations. The class and object-scale distributions of the training, validation and test sets are shown in Figure 1. This study uses 4517 images for training and 565 images for validation during training, structural comparisons, ablation experiments, parameter-sensitivity analysis, and query-behavior analysis. A further 566 images constitute the test set used only for the final robustness evaluation. The network input size is fixed at 384×384 pixels. In the absence of external-dataset validation, the conclusions are limited to CottonWeedDet12 and the present experimental protocol.

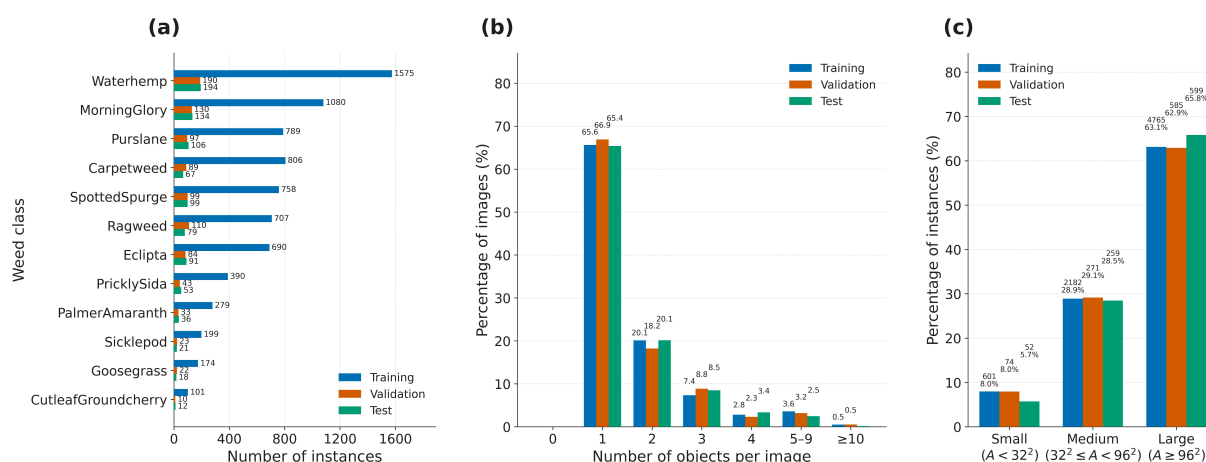


Figure 1. Class, per-image object-count, and object-scale distributions of the training, validation, and test sets. There are 7548, 930, and 910 annotation boxes in the training, validation, and test sets, respectively. (a) Distribution of annotation boxes for each category; (b) statistics of annotation boxes per image; (c) scale-based statistics of annotation boxes.

To mitigate class imbalance, RT-DETRv2, B_Lite, and CQ-RT-DETR use the same repeated-sampling sequence for images containing underrepresented classes, increasing the number of image samples processed per epoch from 4517 to 5004. Based on the resampled image sequence, the cumulative number of bounding-box occurrences per epoch is 8492 (Table 1). YOLO and D-FINE follow the sampling and optimization strategies of their respective official frameworks. The original distributions of the validation and test sets remain unchanged. This procedure generates neither new images nor new annotations; therefore, the numbers of unique images and unique annotations remain unchanged.

Table 1. Comparison of the training set before and after class-balanced sampling.

Data Partition	Number of Unique Images	Samples per Epoch	Number of Unique Annotations	Cumulative Bounding Boxes per Epoch
Original training set	4517	4517	7548	7548
After class-balanced sampling	4517	5004	7548	8492

2.2. Architectural Overview of RT-DETRv2

This study uses RT-DETRv2 with a PResNet18vd backbone as the baseline detector. RT-DETRv2 retains the overall architecture of RT-DETR while improving multiscale sampling, data augmentation, and scale-adaptive hyperparameter configurations [27,28]. The overall architecture is illustrated in Figure 2. Given an input image I , the backbone successively generates feature maps S_3 , S_4 , and S_5 at three scales; their spatial resolutions progressively decrease, while their levels of semantic abstraction increase. The efficient hybrid encoder comprises an attention-based intra-scale feature interaction (AIFI) module and a CNN-based cross-scale feature fusion (CCFF) module. The highest-level feature S_5 first undergoes channel projection and positional encoding and is then fed into AIFI for global semantic interaction, yielding the enhanced high-level feature F_5 , as described by Equations (1) and (2):

$$Q = K = V = \text{Flatten}(S_5) \quad (1)$$

$$F_5 = \text{Reshape}(\text{AIFI}(Q, K, V)) \quad (2)$$

Subsequently, S_3 , S_4 , and F_5 are fed into CCFF, which generates encoded features at three scales through top-down and bottom-up convolutional fusion pathways, as expressed in Equation (3):

$$O = CCFF(S3, S4, F5) \quad (3)$$

After the encoder outputs are flattened, they are represented as N_e candidate locations, with p_n and b_n denoting the class confidence score and predicted bounding box of the n -th candidate location, respectively. The encoder first performs classification and bounding-box regression and then selects high-quality candidate locations as the initial content queries and reference boxes for the decoder. The query-selection process can be formulated as Equation (4):

$$Q = \text{TopK}(\{q_n\}_{n=1}^{N_e}, N_q) \quad (4)$$

Here, q_n denotes the candidate query constructed from the n -th encoded location, and N_q denotes the number of queries fed into the decoder. Within the multilayer Transformer decoder, the selected queries interact with the encoded features through cross-attention, progressively updating the class predictions and bounding-box coordinates layer by layer. During training, one-to-one Hungarian matching is used to assign supervision; during inference, the class predictions and bounding boxes are output directly.

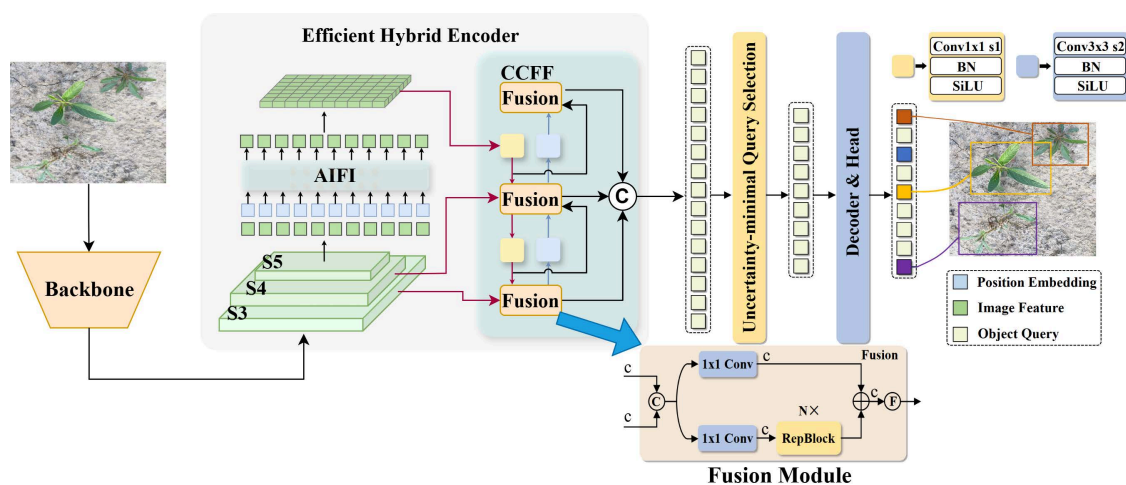


Figure 2. Overall architecture of RT-DETR. RT-DETRv2 retains this architecture while improving multiscale sampling, data augmentation, and training configurations.

2.3. B_Lite: Structural Compression

In the original RT-DETRv2, S3, S4, and S5 are first projected to a common hidden dimension d , around which the AIFI and CCFF output channels, query-selection features, and decoder embeddings are constructed. The number of ground-truth objects in close-range cotton-field images is substantially smaller than the 300 object queries retained by the standard model. Accordingly, the hidden dimension is uniformly reduced from 256 to 192 and the number of object queries from 300 to 100, while the number of decoder layers remains at the default value of three, yielding the lightweight B_Lite baseline shown in Figure 3. For linear projections and 1×1 convolutions whose dominant computational costs scale approximately with d^2 , the theoretical compression ratio of these components is calculated using Equation (5):

$$R_d = 1 - \left(\frac{d_L}{d_B}\right)^2 = 1 - \left(\frac{192}{256}\right)^2 = 43.75\% \quad (5)$$

Here, d_B and d_L denote the hidden dimensions of the baseline and lightweight models, respectively.

The compression ratio of the object-query budget is given by Equation (6):

$$R_q = 1 - \frac{N_{q,L}}{N_{q,B}} = 1 - \frac{100}{300} = 66.67\% \quad (6)$$

The actual parameter count is obtained by counting all network parameters in the inference model, excluding optimizer states, exponential moving average (EMA) weight copies, and runtime caches. GFLOPs are measured using THOP v2.0.20 with a single FP32 input tensor of size $1 \times 3 \times 384 \times 384$, considering only the network forward pass and excluding image preprocessing, predicted-box postprocessing, non-maximum suppression (NMS), and COCO metric computation, with 1 MAC converted to 2 FLOPs. The final parameter counts and GFLOPs are reported in Section 3.

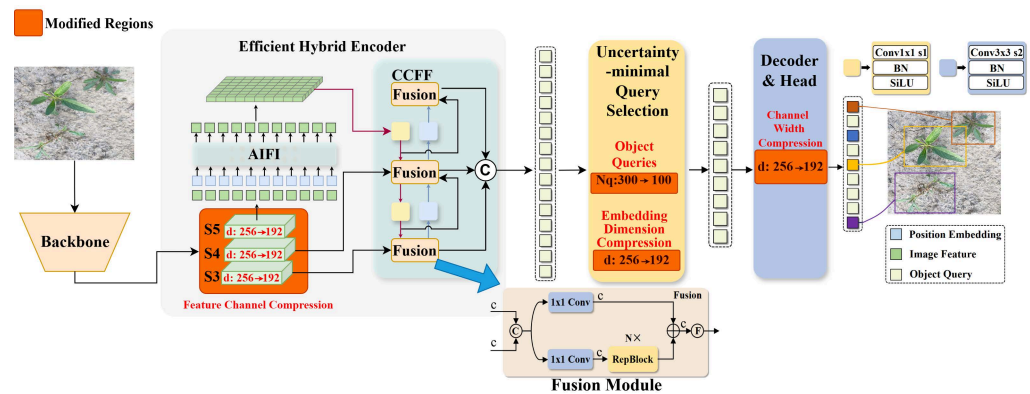


Figure 3. Lightweight architecture of B_Lite. The orange regions indicate where the hidden dimension and number of object queries are modified.

2.4. SR-TCCM: Scale-Reliability-Guided Trajectory-Consistent Competition Margin Calibration

After compressing the inference architecture, no additional inference modules are introduced into the encoder, decoder, or detection head; instead, competition among queries associated with the same ground-truth object is calibrated during training. SR-TCCM uses the one-to-one Hungarian matching results from the final decoder layer as anchors and constrains only the relationship between the class scores of the matched winner query and unmatched competing queries; it neither alters the original matching assignments nor converts the competing queries into additional positive samples. Figure 4 presents the overall architecture of CQ-RT-DETR, in which the lower orange branch represents SR-TCCM and is activated only during training.

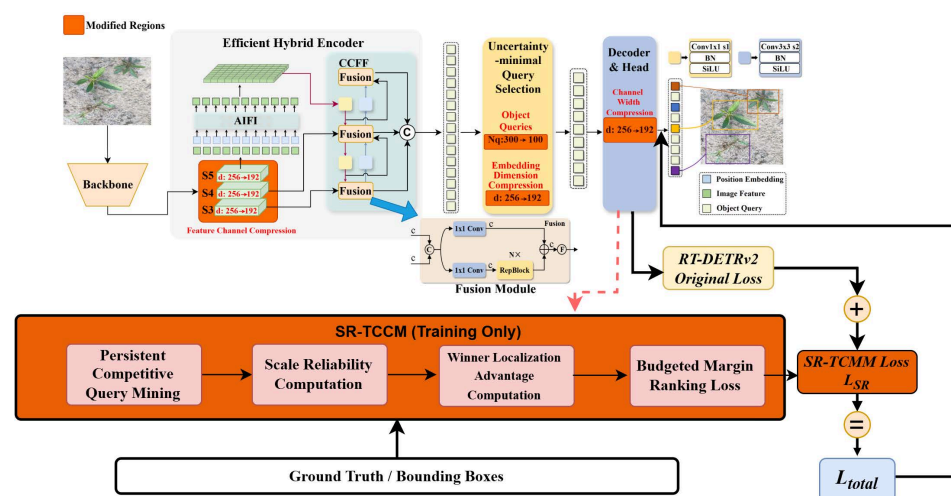


Figure 4. Network architecture of CQ-RT-DETR. The lower orange SR-TCCM branch is activated only during training.

2.4.1. Persistent Competitive Query Mining

The objective of this stage is to construct, for each ground-truth object, a candidate pool free from cross-object interference, thereby identifying which unmatched queries may compete with the winner query, as illustrated in Figure 5. Let the decoder comprise L layers, and let $b_i^{(l)}$ denote the predicted bounding box of query i at decoder layer l ; the j -th ground-truth box in the current image is denoted by g_j . The cross-layer overlap trajectory of query i with respect to target j is defined in Equation (7):

$$u_{ij}^{(l)} = \text{IoU}(b_i^{(l)}, g_j), \quad l = 1, \dots, L. \quad (7)$$

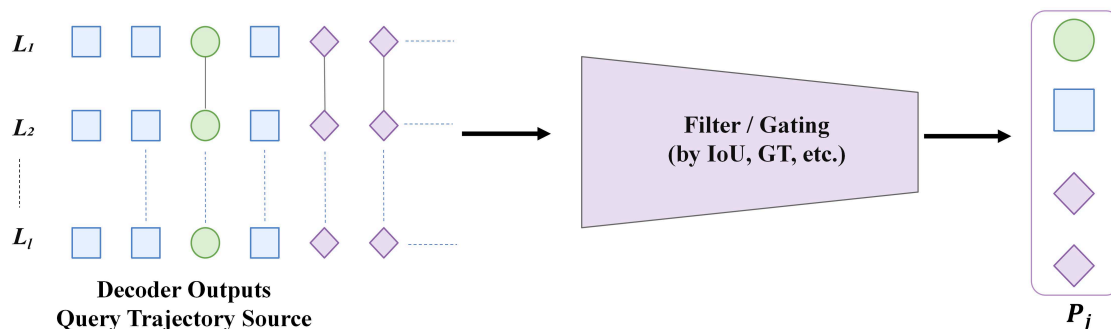


Figure 5. Schematic illustration of persistent competing-query mining.

Suppose that the current image contains N_g ground-truth objects. The index of the query matched to g_j at the final decoder layer is denoted by the winner query w_j ; $\mathcal{M}$ denotes the set of query indices matched to any ground-truth object at the final layer; and $p_{i,c_j}^{(L)}$ denotes the final-layer probability assigned by query i to the ground-truth class c_j . The initial candidate pool for target g_j is defined in Equation (8):

$$P_j = \left\{ i \notin \mathcal{M} \mid \underset{1 \leq k \leq N_g}{\operatorname{argmax}} u_{ik}^{(L)} = j, u_{ij}^{(L)} \geq \theta_u, p_{i,c_j}^{(L)} \geq \theta_p, u_{ij}^{(L)} \leq u_{w_j}^{(L)} \right\} \quad (8)$$

where $\theta_u = 0.25$ and $\theta_p = 0.03$. The five conditions in Equation (8), read from left to right, are denoted as C1–C5. Condition C1, $i \notin \mathcal{M}$, excludes queries already matched to and supervised by any ground-truth object. Condition C2, $\underset{1 \leq k \leq N_g}{\operatorname{argmax}} u_{ik}^{(L)} = j$, requires g_j to be the ground-truth object with the highest final-layer IoU for query i , thereby avoiding cross-object interference. Condition C3, $u_{ij}^{(L)} \geq \theta_u$, removes candidates with insufficient localization overlap. Condition C4, $p_{i,c_j}^{(L)} \geq \theta_p$, removes candidates with a weak classification response to the ground-truth class. Condition C5, $u_{ij}^{(L)} \leq u_{w_j}^{(L)}$, prevents a query with better final-layer localization than the winner query from being treated as a suppression target.

This stage constructs only a permissive candidate pool P_j and does not immediately apply a ranking loss, thereby separating candidate-eligibility assessment from the confirmation of persistent competition in the next stage.

2.4.2. Cross-Layer Trajectory Stability and Persistent Competing Query Selection

The set P_j obtained in Stage 1 may still contain queries that happen to approach the target only at the final decoder layer. This stage uses the full IoU trajectory defined in Equation (7) to determine whether a candidate query remains associated with the same target across multiple decoder layers, as illustrated in Figure 6. Trajectory stability is first defined in Equation (9):

$$s_{ij} = \exp \left[-\frac{\operatorname{Var}_{l=1}^L (u_{ij}^{(l)})}{\tau_s} \right], \quad \tau_s = 0.03, i \in P_j \quad (9)$$

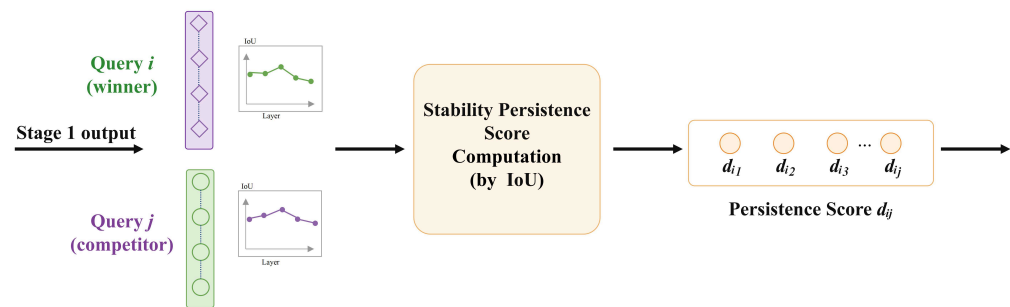


Figure 6. Schematic illustration of cross-layer trajectory stability and competitive persistence.

The variance is calculated as the population variance across the L decoder layers. Smaller trajectory fluctuations yield a value of s_{ij} closer to 1; if a query approaches the target only briefly at a few layers, its stability decreases substantially. Next, μ_{ij} denotes the mean overlap across decoder layers and is combined with the final-layer IoU and trajectory stability to measure competition persistence, as defined in Equations (10)–(12):

$$\mu_{ij} = \frac{1}{L} \sum_{l=1}^L u_{ij}^{(l)}, \quad (10)$$

$$d_{ij} = u_{ij}^{(L)} s_{ij} (0.5 + 0.5\mu_{ij}), \quad (11)$$

$$C_j = \text{TopK}_K (\mathcal{D}_j; d_{ij}), \quad K = 2. \quad (12)$$

The score d_{ij} jointly reflects final localization quality, cross-layer trajectory stability, and the mean overlap throughout the decoding process. Candidate queries are ranked in descending order of d_{ij} , and at most $K = 2$ queries are retained for each ground-truth object to form the final persistent competition set C_j . Therefore, the output of this stage is no longer a permissive candidate pool, but query pairs (w_j, i) supported by explicit cross-layer evidence and their persistence scores d_{ij} ; together, these serve as the inputs to Stages 3 and 4.

2.4.3. Scale Reliability and Winner Localization Advantage

Stage 2 determines which queries remain in persistent competition, but the reliability of different objects and winner queries varies. Small objects are more sensitive to displacements of only a few pixels, and applying a uniform competition strength may amplify their IoU fluctuations; moreover, when the winner query is only marginally better localized than a competing query, forcibly widening their score gap may reinforce an unstable match. Accordingly, this stage separately calculates object-scale reliability and winner localization advantage and combines them with d_{ij} to obtain a query-pair weight, as illustrated in Figure 7.

Let the normalized width and height of ground-truth box g_j be w_j^s and h_j^s , respectively, giving the normalized area $a_j = w_j^s h_j^s$; a_{ref} denotes the median normalized area of the ground-truth boxes in the training set. Scale reliability is defined in Equation (13):

$$r_j^{\text{scale}} = r_{\min} + (1 - r_{\min}) \min \left(1, \sqrt{\frac{a_j}{a_{\text{ref}}}} \right), \quad r_{\min} = 0.10. \quad (13)$$

The square-root mapping slows the decrease in weight as the object area becomes smaller, allowing small objects to retain a minimum level of competition supervision while the weights of medium and large objects smoothly approach 1. This term depends only on the ground-truth box scale and does not fluctuate with the current predictions during training.

For $i \in C_j$, the winner localization advantage is calculated from the final-layer IoU difference, as defined in Equation (14):

$$g_{ij}^{\text{win}} = \sigma \left(\frac{u_{wj}^{(L)} - u_{ij}^{(L)}}{\tau_w} \right), \tau_w = 0.05 \quad (14)$$

where $\sigma(\cdot)$ is the sigmoid function. When the winner query and competing query have similar localization quality, g_{ij}^{win} approaches 0.5 and the ranking constraint is weakened; when the winner query has a clear localization advantage, the gate gradually approaches 1. The final outputs of Stage 3 are calculated using Equations (15) and (16):

$$\omega_{ij} = d_{ij} r_j^{\text{scale}} g_{ij}^{\text{win}}, \quad (15)$$

$$m_{ij} = m_0 + \lambda_m d_{ij}, m_0 = 0.02, \lambda_m = 0.10. \quad (16)$$

Here, ω_{ij} determines the contribution of the query pair to the ranking loss, whereas m_{ij} adaptively specifies the required score margin according to the persistence of the competition. Thus, the persistence score d_{ij} produced by Stage 2 is transformed into a weight–margin pair that can be used directly in Stage 4: greater persistence, higher scale reliability, and a clearer winner advantage result in a stronger ranking constraint in the subsequent stage.

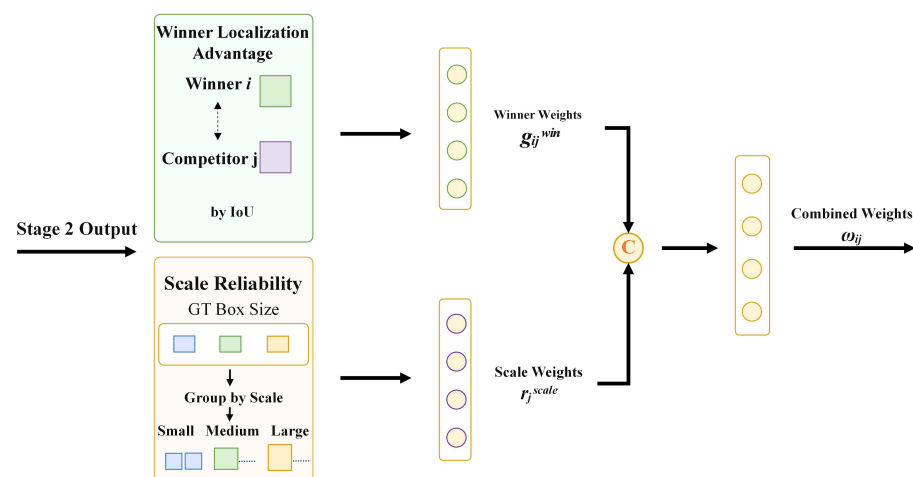


Figure 7. Schematic illustration of scale reliability and winner localization advantage.

2.4.4. Budgeted Adaptive Margin Ranking Loss

Stage 4 receives the final competition set C_j , query-pair weights ω_{ij} , and adaptive margins m_{ij} . Its objective is to make the logit of a reliable winner query for the ground-truth class higher than that of a persistent competing query associated with the same target, without altering the original one-to-one definition of positive and negative samples, as illustrated in Figure 8. Let the final-layer logits of the winner query and competing query for class c_j be z_{w_j, c_j} and z_{i, c_j} , respectively. Equation (17) defines the temperature-smoothed, weighted margin-ranking loss:

$$\mathcal{L}_{\text{rank}} = \frac{\sum_{j \in G} \sum_{i \in C_j} \omega_{ij} T \log \left[1 + \exp \left(\frac{z_{i, c_j} - z_{w_j, c_j} + m_{ij}}{T} \right) \right]}{\max \left(\sum_{j \in G} \sum_{i \in C_j} \omega_{ij}, \varepsilon \right)}, T = 0.20, \varepsilon = 10^{-6} \quad (17)$$

where G denotes the index set of ground-truth objects in the current batch, and $\varepsilon = 10^{-6}$ is a numerical-stability constant. In the implementation, the normalization denominator is clamped to a minimum of ε ; when no valid competing query exists, the ranking loss is set to zero. The loss compares only the winner query and persistent competing queries associated

with the same ground-truth object. It neither relabels competing queries as positive samples nor alters the Hungarian matching results; its purpose is to improve query-score ranking rather than introduce an additional detection branch.

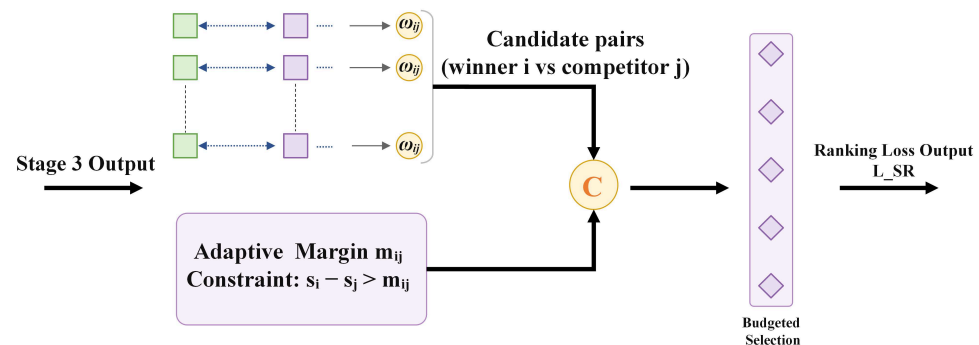


Figure 8. Schematic illustration of the budget-constrained adaptive margin ranking loss.

To avoid introducing excessively strong supervision before the matching relationships stabilize early in training, the activation epoch is set to $e_0 = 30$ and the linear ramp-up period to $E_r = 20$. The scheduling coefficient is defined in Equation (18):

$$\lambda(e) = \begin{cases} 0, & e < e_0, \\ \min\left(1, \frac{e-e_0+1}{E_r}\right), & e \geq e_0. \end{cases} \quad (18)$$

Let $\eta = 0.10$ denote the ranking-loss coefficient, and let $\kappa = 0.03$ denote the budget ratio relative to the Varifocal Loss (VFL) [33] at the final decoder layer. The unclipped auxiliary term, budget cap, and final SR-TCCM loss are given in Equations (19), (20), and (21), respectively:

$$\tilde{\mathcal{L}}_{SR} = \eta \lambda(e) \mathcal{L}_{\text{rank}}, \quad (19)$$

$$B(e) = \kappa \lambda(e) \mathcal{L}_{\text{vfl}}^{\text{final}}, \quad (20)$$

$$\mathcal{L}_{SR} = \min\left(\tilde{\mathcal{L}}_{SR}, B(e)\right). \quad (21)$$

The budget constraint ensures that the additional ranking term never exceeds 3% of the final-layer Varifocal Loss at any stage of training, thereby preventing the ranking objective from dominating the original classification and localization optimization. When the budget cap is calculated, gradient propagation through the VFL value used as the scale reference is stopped so that the auxiliary branch cannot alter the optimization direction of the original loss through the budget calculation. The final training objective is defined in Equation (22):

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RT-DETRv2}} + \mathcal{L}_{SR}. \quad (22)$$

In summary, Stage 1 produces the candidate pool P_i ; Stage 2 refines it into the persistent competition set C_i and yields d_{ij} ; Stage 3 combines competition persistence, scale reliability, and winner advantage to obtain ω_{ij} and m_{ij} ; and Stage 4 produces the budget-constrained loss $\mathcal{L}_{SR}$. SR-TCCM does not modify the hybrid encoder, Transformer decoder, prediction heads, Hungarian matching, or postprocessing. The branch is completely removed after training; therefore, CQ-RT-DETR and B_Lite have identical inference parameter counts, GFLOPs, and ONNX inference computation graphs.

2.5. Training Configuration and Evaluation Metrics

RT-DETRv2, B_Lite, and CQ-RT-DETR use the same data partition, 384×384 input size, 80 training epochs, batch sizes, repeated-sampling sequence, optimizer, learning-rate schedule, and data augmentation. Cross-framework comparisons standardize only the data

partition, input size, and training duration; pretraining, sampling, optimization, learning-rate scheduling, and data augmentation for YOLO and D-FINE follow their respective official implementations. The RT-DETR-series configuration is provided in Table 2, and the cross-framework training protocols are provided in Supplementary Table S1. Except for the final robustness evaluation, the structural comparisons, ablation experiments, parameter-sensitivity analysis, and query-behavior analysis use a fixed random seed of 42 and are evaluated on the validation set. The final robustness evaluation uses five random seeds; the best weights for each model are selected on the validation set and then evaluated only on the test set. The test set is not used for model selection or parameter tuning. Table 3 lists the SR-TCCM training parameters to facilitate reproducibility.

Table 2. Main experimental environment and training configuration for the RT-DETR series.

Item	Setting
Hardware	NVIDIA RTX 3080 Ti 12 GB; Intel Xeon Silver 4214R CPU; 90 GB RAM
Software framework	Ubuntu 22.04; PyTorch 2.8.0 + cu128; RT-DETRv2 PyTorch
Input resolution	384 × 384
Epochs	80
Batch size	Training: 8; validation: 16
Optimizer	AdamW
Initial learning rate	1×10^{-4}
Mixed precision	AMP
Random seeds	Primary experiments: 42; final robustness evaluation: 42, 3407, 5903, 45, and 46
Primary metrics	COCO AP50–95, AP50, AP75, APs, APm, APi, and AR
Weight decay	AdamW: 1×10^{-4} ; Norm/BN: 0; beta = (0.9, 0.999)
Learning-rate schedule	LinearWarmup for the first 2000 iterations, increasing from approximately 1×10^{-7} to 1×10^{-4} , followed by a constant learning rate of 1×10^{-4} through epoch 80
Pretrained	True
Data augmentation	PhotometricDistort ($p = 0.5$), fixed resize to 640, RandomZoomOut, RandomIoUCrop ($p = 0.8$), horizontal flipping, and final resize to 384
Multiscale training	None
Decoder configuration	B_Lite/CQ: 3 decoder layers; 100 queries
Original detection-loss weights	VFL:L1 bbox:GIoU = 1:5:2
Hungarian-matcher weights	class:(bbox:GIoU = 2:5:2

The SR-TCCM hyperparameters in Table 3 are fixed before training. Conservative values are selected on the basis of method-development experience, the ranges of the relevant variables, and the principle of weak intervention; no exhaustive search is performed. The quantities that vary dynamically during training are sample-dependent weights, adaptive margins, and the actual loss cap, rather than the hyperparameters listed in the table. All parameter comparisons are conducted on the validation set, and the test set is not involved in parameter selection.

The COCO evaluation protocol is adopted [34]. AP50–95 is the mean AP over IoU thresholds from 0.50 to 0.95 at intervals of 0.05:

$$AP_{50-95} = \frac{1}{10} \sum_{k=0}^9 AP_{0.50+0.05k} \quad (23)$$

For the final five-seed robustness evaluation, the best checkpoint for each model was selected on the validation set and evaluated once on the same test set.

Table 3. Training parameters of SR-TCCM.

Parameter	Description	Value
θ_u	Minimum final-layer IoU for competing candidates	0.25
θ_p	Minimum probability for the ground-truth class	0.03
τ_s	Trajectory-stability temperature	0.03
K	Maximum number of persistent competing queries per ground-truth object	2
r_{min}	Lower bound of scale reliability	0.1
τ_w	Winner-localization-advantage temperature	0.05
m_0	Base margin	0.02
λ_m	Adaptive-margin increment coefficient	0.1
T	Margin-ranking temperature	0.2
e_0	SR-TCCM activation epoch	30
E_r	Linear ramp-up period	20
η	Ranking-loss coefficient	0.1
κ	VFL-budget ratio	0.03

2.6. RKNN Deployment and Testing

Because both structural compression and query calibration are designed specifically for RT-DETRv2, the board-level deployment experiment compares only RT-DETRv2, B_Lite, and CQ-RT-DETR to examine the speed changes from the original model to the structurally compressed model and the final model. All three models are exported as ONNX models with a fixed input shape of $1 \times 3 \times 384 \times 384$ [35] and converted using RKNN-Toolkit2 2.3.2 into FP16 RKNN models for the RK3576 platform. The YOLO series is used only for cross-model detection-performance comparisons and is not included in the board-level deployment experiment.

Testing is performed on a DC-A576 development board equipped with an RK3576, running Ubuntu 22.04 with RKNN Runtime/RKNNLite2 2.3.2 and RKNPU driver 0.9.8. The tests use a 384×384 input, a batch size of 1, and both NPU cores. After 50 warm-up runs, each model performs 300 consecutive inferences using cached input, from which mean forward-pass latency and FPS are calculated. Timing covers only model forward execution and excludes image loading, preprocessing, postprocessing, communication, and actuator response. Detection accuracy is calculated with the original PyTorch FP32 weights on the 566-image test set using COCO eval; the board-level FP16 models are used only for speed testing, and AP is not recalculated for them.

3. Results

3.1. Structural Compression Results and Scale-Specific Effects

After the hidden dimension d of the original RT-DETRv2 is reduced from 256 to 192 and the number of object queries from 300 to 100, the parameter count of B_Lite decreases from 20.097 M to 16.694 M (a reduction of 16.93%) while GFLOPs decrease from 22.947 to 17.581 (a reduction of 23.38%). Meanwhile, AP50–95 decreases from 0.8799 to 0.8684, corresponding to an absolute decrease of 0.0115 (1.15 percentage points; a relative decrease of 1.31%). Table 4 summarizes the principal metrics before and after structural compression.

To further examine the effect of structural compression on objects at different scales, Table 5 presents the scale-specific results for B_Lite and the original RT-DETRv2. The relative AP decreases in B_Lite for medium and large objects are only 4.62% and 1.29%, respectively, whereas small-object AP decreased from 0.267 to 0.161, a relative reduction

of 39.70%, indicating that small objects are more sensitive to compression of the channel width and query budget. Notably, small-object AR increased from 0.573 to 0.644 despite the pronounced decrease in AP. This pattern suggests that the model continued to cover some small-object candidates, but their localization quality or confidence ranking may have become unstable.

Table 4. Model size, computational cost, and accuracy before and after structural compression.

Model	RT-DETRv2	B_Lite	Change
Hidden dimension	256	192	−25%
Number of object queries	300	100	−66.67%
Trainable parameters (M)	20.097	16.694	−16.93%
GFLOPs	22.947	17.581	−23.38%
AP50–95	0.8799	0.8684	−0.0115 (−1.31%)

Note: Results were evaluated on the validation set of 565 images.

Table 5. Scale-specific performance changes in B_Lite relative to RT-DETRv2.

Metric	RT-DETRv2	B_Lite	Change
AP50–95@small	0.267	0.161	−39.70%
AP50–95@medium	0.628	0.599	−4.62%
AP50–95@large	0.932	0.920	−1.29%
AR50–95@small	0.573	0.644	12.39%
AR50–95@medium	0.858	0.840	−2.10%
AR50–95@large	0.965	0.951	−1.45%

Note: Results were evaluated on the validation set of 565 images.

3.2. Accuracy Recovery with SR-TCCM

After the training-only SR-TCCM mechanism was added to B_Lite, the parameter count and GFLOPs of CQ-RT-DETR remained unchanged at 16.694 M and 17.581, respectively, while AP50–95 increased from 0.8684 to 0.8842. This represents an absolute gain of 1.58 percentage points (a relative improvement of 1.82%); compared with the original RT-DETRv2, AP50–95 increased by 0.43 percentage points (a relative improvement of 0.49%). As shown in Table 6, SR-TCCM recovered the accuracy lost through structural compression without increasing inference-graph complexity.

Table 6. Overall performance before and after introducing SR-TCCM.

Model	RT-DETRv2	B_Lite	With SR-TCCM	Change (vs. RT-DETRv2)	Change (vs. B_Lite)
Hidden dimension	256	192	192	−25%	/
Number of object queries	300	100	100	−66.67%	/
Trainable parameters (M)	20.097	16.694	16.694	−16.93%	/
GFLOPs@384	22.947	17.581	17.581	−23.38%	/
AP50–95	0.8799	0.8684	0.8842	+0.0043 (+0.49%)	+0.0158 (+1.82%)

Note: Results were evaluated on the validation set of 565 images.

The scale-specific results are presented in Table 7. With the fixed random seed of 42 on the validation set, CQ-RT-DETR increases small-, medium-, and large-object AP from 0.161, 0.599, and 0.920 for B_Lite to 0.282, 0.647, and 0.930, respectively. Small-object AR

decreases from 0.644 to 0.589 but remains above the 0.573 obtained by the original model. This single validation run is used to describe the direction of change rather than to support a claim of statistical significance; the scale-specific means and standard deviations from the five-seed evaluation on the test set are reported in Section 3.7.

Table 7. Scale-specific performance before and after introducing SR-TCCM.

Metric	RT-DETRv2	B_Lite	CQ-RT-DETR
AP50–95@small	0.267	0.161	0.282
AP50–95@medium	0.628	0.599	0.647
AP50–95@large	0.932	0.920	0.930
AR50–95@small	0.573	0.644	0.589
AR50–95@medium	0.858	0.840	0.855
AR50–95@large	0.965	0.951	0.956

Note: Results were evaluated on the validation set of 565 images.

3.3. Ablation Study

To distinguish the effects of the two structural-compression factors, a 2×2 ablation experiment is conducted. As shown in Table 8, when the hidden dimension remains at 256, reducing the number of object queries from 300 to 100 decreases AP50–95 by 0.0046 and GFLOPs by 4.90%, while leaving the parameter count unchanged. When the number of queries remains at 300, reducing the hidden dimension from 256 to 192 decreases AP50–95 by 0.0083 and reduces the parameter count and GFLOPs by 16.93% and 19.03%, respectively. Joint compression in B_Lite reduces the parameter count from 20.097 M to 16.694 M and GFLOPs from 22.947 to 17.581, while AP50–95 decreases from 0.8799 to 0.8684. These results indicate that hidden-dimension compression is the main source of both the complexity reduction and the initial accuracy loss, whereas query-budget compression has a smaller effect on overall accuracy and primarily provides an additional reduction in decoder computation. No pronounced compounding degradation is observed when the two compression operations are combined, providing a lightweight basis on which SR-TCCM can subsequently recover the compression-induced accuracy loss.

Table 8. Structural ablation of the hidden dimension and query budget.

Hidden Dimension	Object Queries	Parameters (M)	GFLOPs	AP50–95
256	300	20.097	22.947	0.8799
256	100	20.097	21.721	0.8753
192	300	16.694	18.495	0.8716
192	100	16.694	17.581	0.8684

As shown in Table 9, the complete CQ-RT-DETR achieves the highest AP50–95 (0.8842). With all other settings held constant, removing trajectory screening, scale reliability, or the winner localization advantage decreases AP50–95 by 0.0070, 0.0052, and 0.0043, respectively. Replacing the adaptive margin with a fixed margin or removing the loss budget decreases AP50–95 by 0.0024 and 0.0040, respectively. These single-factor validation results are consistent with the intended roles of the components, with trajectory screening and scale reliability showing relatively larger effects.

Table 9. Ablation analysis of SR-TCCM.

Variant	Only Change	AP50–95	APs	APm	API
B_Lite	Without SR-TCCM	0.8684	0.161	0.599	0.920
E1	Trajectory screening removed	0.8772	0.166	0.624	0.927
E2	Scale reliability removed	0.8790	0.170	0.615	0.929
E3	Winner localization advantage removed	0.8799	0.255	0.619	0.929
E4	Fixed margin	0.8818	0.152	0.632	0.930
E5	Loss budget removed	0.8802	0.263	0.623	0.929
CQ-RT-DETR	Complete model	0.8842	0.282	0.647	0.930

Note: Results were evaluated on the validation set of 565 images.

3.4. Query-Competition Behavior Analysis

To analyze the effects of structural compression and SR-TCCM on query behavior, offline statistics are calculated on the fixed validation set. The final-layer Hungarian-matched query is treated as the winner, and the initial competition set P_j and persistent competition set C_j are determined according to Equations (8)–(12). The 95% confidence intervals are estimated from 20,000 paired bootstrap resamples using the ground-truth object as the sampling unit.

As shown in Table 10, the joint structural compression from RT-DETRv2 to B_Lite changes both the hidden dimension and the query budget. The competition occurrence rate subsequently increases from 73.98% to 89.35%, while the numbers of initial and persistent competing queries increase from 5.419 and 1.337 to 7.125 and 1.684, respectively. These changes are consistent with the hypothesis of stronger same-target query competition after compression, but they cannot be attributed to query-budget compression alone. Relative to B_Lite, CQ-RT-DETR reduces the competition occurrence rate to 81.51% and the numbers of initial and persistent competing queries to 3.762 and 1.460, respectively, while increasing the winner–competitor logit gap from 5.585 to 5.933. The ranking-reversal rate changes only slightly, and the rate at which the final-layer winner retains the highest IoU across decoder layers does not increase. Thus, the present evidence mainly supports an association of SR-TCCM with fewer redundant competing queries and stronger separation of class scores.

Table 10. Query-competition behavior statistics.

Metric	RT-DETRv2	B_Lite	CQ-RT-DETR
Number of queries	300	100	100
Competition occurrence rate (%)	73.98 [71.18, 76.77]	89.35 [87.31, 91.29]	81.51 [79.03, 83.98]
Initial competing queries/GT	5.419 [4.958, 5.898]	7.125 [6.605, 7.671]	3.762 [3.523, 4.014]
Persistent competing queries/GT	1.337 [1.281, 1.391]	1.684 [1.641, 1.726]	1.460 [1.410, 1.511]
Final-layer winner retained as top-IoU query across layers (%)	62.08 [59.39, 64.73]	77.26 [74.95, 79.57]	63.55 [60.97, 66.09]
Winner–competitor logit gap	5.715 [5.567, 5.856]	5.585 [5.461, 5.706]	5.933 [5.801, 6.061]
Ranking-reversal rate (%)	1.13 [0.50, 1.85]	1.09 [0.52, 1.72]	1.25 [0.65, 1.95]

Note: Results were evaluated on the validation set of 565 images.

3.5. Sensitivity Analysis of Key Parameters

To determine whether the performance gain depends on fine-grained tuning, eight single-factor sensitivity experiments are conducted for four key parameters. As shown in Table 11, six of the eight alternative settings perform below the default configuration.

Although $\theta_u = 0.20$ and $\tau_s = 0.05$ yield slightly higher values than their defaults, the overall differences are small; $K = 2$ and $\kappa = 0.03$ perform best within their respective parameter groups. These results indicate that SR-TCCM does not depend on a unique, finely searched parameter combination, although the trajectory-screening strength, number of competing queries, and loss budget must remain within reasonable ranges.

Table 11. Sensitivity analysis of four key hyperparameters.

Parameter	Lower Value	Default Value	Higher Value
θ_u	0.20: 0.8874	0.25: 0.8842	0.30: 0.8806
τ_s	0.02: 0.8692	0.03: 0.8842	0.05: 0.8856
K	1: 0.8717	2: 0.8842	3: 0.8771
κ	0.01: 0.8832	0.03: 0.8842	0.05: 0.8775

Note: Results were evaluated on the validation set of 565 images.

3.6. Comparison with Other Detection Models

To compare different detection architectures, YOLOv8n [36], YOLOv8m [36], YOLO11n [37], YOLO11m [37], YOLO26n [38], YOLO26m [38], and D-FINE-M [39] are trained using the same data partition, a 384×384 input, and 80 training epochs, while framework-specific sampling, pretraining, optimization, and augmentation strategies follow their respective official implementations. The results are presented in Table 12. The AP50 of CQ-RT-DETR is 0.9392, only 0.0009 below the highest value of 0.9401 obtained by YOLOv8m; its AP75 and AP50–95 values are 0.9127 and 0.8842, which are 0.0040 and 0.0034 below the corresponding highest values. Its parameter counts and GFLOPs are 35.4% and 38.0% lower than those of YOLOv8m and 18.0% and 28.1% lower than those of YOLO26m, respectively. Compared with the original RT-DETRv2, CQ-RT-DETR maintains comparable accuracy while using fewer parameters and GFLOPs. Thus, under the present protocol, CQ-RT-DETR provides a favorable accuracy–complexity trade-off, but the results do not imply that it is more accurate than every comparison model. Published studies on CottonWeedDet12 use different data partitions, input sizes, training epochs, augmentation strategies, and evaluation protocols; their results therefore cannot be ranked directly against those of the present study [40]. These published results are summarized in Supplementary Table S2 for background reference only.

Table 12. Performance comparison among different detection models.

Model	AP50	AP75	AP50–95	Parameters (M)	GFLOPs
YOLOv8n	0.918	0.893	0.859	3.008	2.915
YOLOv8m	0.9401	0.9138	0.8866	25.847	28.341
YOLO11n	0.903	0.867	0.838	2.584	2.277
YOLO11m	0.9351	0.9167	0.8841	20.039	24.371
YOLO26n	0.9104	0.8847	0.8459	2.377	1.872
YOLO26m	0.9390	0.9165	0.8876	20.359	24.442
D-FINE-M	0.9371	0.9123	0.8823	22.595	19.485
RT-DETRv2	0.9354	0.9121	0.8799	20.097	22.947
CQ-RT-DETR	0.9392	0.9127	0.8842	16.694	17.581

Note: Results were evaluated on the validation set of 565 images.

3.7. Robustness Analysis

To examine robustness to random initialization, RT-DETRv2, B_Lite, and CQ-RT-DETR are trained under the same settings using the same five random seeds: 42, 3407, 5903, 45, and 46. As shown in Table 13, structural compression reduces mean AP50–95 from 0.8867 to 0.8691; after SR-TCCM is introduced, CQ-RT-DETR reaches 0.8892, and its improvement over B_Lite is consistent across all five seeds. Because the difference between CQ-RT-DETR and RT-DETRv2 is not significant, the primary contribution lies in recovering the compression-induced accuracy loss and improving the accuracy–efficiency trade-off.

Table 13. Performance of different models across five random seeds.

Model	AP50–95	AP (Small)	AP (Medium)	AP (Large)
RT-DETRv2	0.8867 ± 0.0032	0.2709 ± 0.0236	0.6323 ± 0.0302	0.9266 ± 0.0024
B_Lite	0.8691 ± 0.0030	0.1625 ± 0.0312	0.6145 ± 0.0131	0.9206 ± 0.0014
CQ-RT-DETR	0.8892 ± 0.0044	0.2733 ± 0.0432	0.6152 ± 0.0120	0.9296 ± 0.0041

Note: Values are the mean ± standard deviation across five random seeds on the test set of 566 images. CQ-RT-DETR improves mean AP50–95 by 2.012 percentage points over B_Lite (95% CI: [1.411, 2.613], $p = 0.0007$) and by 0.255 percentage points over RT-DETRv2; the latter difference is not significant ($p = 0.4127$).

To analyze class-specific performance, detection results are calculated for all 12 weed classes, as shown in Figure 9; detailed values are provided in Supplementary Tables S3 and S4. Relative to B_Lite, CQ-RT-DETR improves AP50–95 for 7 classes and AR100 for 9 classes, with AP50–95 gains of 0.1332 and 0.0729 for Carpetweed and Ragweed, respectively. Relative to RT-DETRv2, CQ-RT-DETR provides broadly comparable class-specific AP, while AR100 increases for all 12 classes. Because some classes contain relatively few ground-truth boxes in the test set, the class-specific results may still exhibit substantial variability.

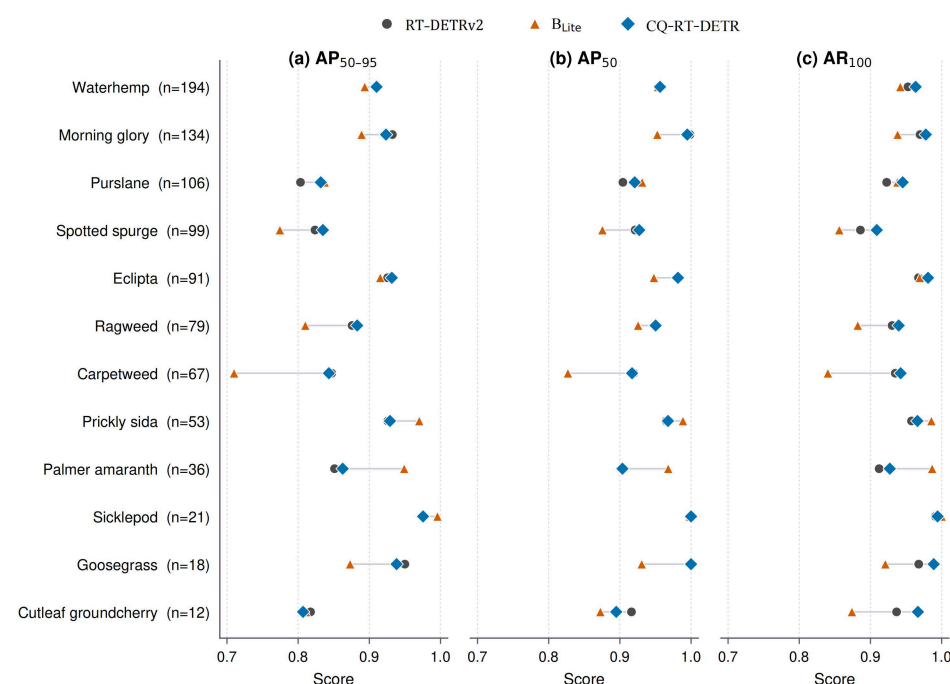


Figure 9. Class-wise detection performance of the three models on the test set. Results are averaged over five random seeds, and n denotes the number of ground-truth bounding boxes for each class.

3.8. RKNN Deployment Performance

Table 14 reports the model-level mean forward-pass latency, FPS, and model size of RT-DETRv2, B_Lite, and CQ-RT-DETR on the RK3576 platform. Relative to RT-DETRv2, CQ-RT-DETR reduces mean forward-pass latency by 54.57%, while increasing FPS from 6.69 to 14.7. CQ-RT-DETR and B_Lite have nearly identical speeds, confirming that SR-TCCM introduces no inference-time overhead.

Table 14. Deployment performance of the three RT-DETRv2-series models on the RK3576 platform.

Model	Mean Forward-Pass Latency (ms)	FPS	Model Size (MB)
RT-DETRv2	149.44	6.69	45.56
B_Lite	67.56	14.8	38.39
CQ-RT-DETR	67.89	14.7	38.39

Note: The development board is operated in performance mode, and both RK3576 NPU cores are invoked using NPU_CORE_ALL. The input size is 384×384 with a batch size of 1. Each model is warmed up for 50 runs and then executed continuously for 300 inferences. The reported latency covers only model forward execution.

4. Discussion

The performance degradation after structural compression cannot be attributed to a single factor. Across the five-seed evaluation on the test set, mean AP_{50–95} decreases from 0.8867 for RT-DETRv2 to 0.8691 for B_Lite, while small-object AP decreases from 0.2709 to 0.1625 and the changes for medium and large objects are smaller. Together with the 2×2 ablation results, these findings indicate that hidden dimension compression is the main source of the accuracy loss, whereas query-budget compression has a smaller effect. The query-behavior statistics further show that the competition occurrence rate and the number of competing queries increase after joint compression. These observations are consistent with the hypothesis of intensified query competition, but they may also be influenced by reduced feature-representation capacity, class imbalance, and annotation noise.

SR-TCCM addresses the above hypothesis through training-time calibration. Across the five-seed evaluation on the test set, it increases the mean AP_{50–95} of B_Lite from 0.8691 to 0.8892 and small-object AP from 0.1625 to 0.2733, while maintaining overall accuracy comparable to that of RT-DETRv2. Query-behavior analysis further associates CQ-RT-DETR with a lower competition occurrence rate, fewer persistent competing queries, and a larger logit gap. The ranking-reversal rate changes only slightly, whereas the cross-layer highest-IoU retention rate decreases rather than increases. Thus, the current evidence supports an association of SR-TCCM with accuracy recovery in the compressed model and improved query-score separation, but it is insufficient to establish query competition as the sole mechanism.

This study has several limitations. First, the experiments are based only on Cotton-WeedDet12, and generalization across regions, imaging devices, and crop scenarios requires validation on external datasets. Second, the deployment results are forward-pass benchmarks for FP16 RKNN models on the RK3576 platform; they exclude image acquisition, preprocessing, postprocessing, communication, and actuator response and therefore cannot be interpreted as the end-to-end latency of a complete spraying system. Third, SR-TCCM depends on the winner query produced by Hungarian matching. When matching is unstable early in training or annotation noise is substantial, excessively strong ranking supervision may be detrimental; delayed activation and a loss budget are therefore used to constrain its influence. The five-seed experiments and query-behavior statistics improve the credi-

bility of the results but remain correlational evidence from a single dataset and hardware platform. Future work should evaluate additional datasets, compression strengths, and hardware platforms.

5. Conclusions

This study proposes CQ-RT-DETR for multiclass weed detection in natural cotton fields. In the first stage, the hidden dimension of RT-DETRv2 is reduced from 256 to 192 and the number of object queries from 300 to 100, decreasing the parameter count from 20.097 M to 16.694 M and GFLOPs from 22.947 to 17.581. Across five random seeds on the test set, structural compression decreases mean AP_{50–95} from 0.8867 to 0.8691 and small-object AP from 0.2709 to 0.1625, showing that the efficiency gain is accompanied by a measurable loss of accuracy.

In the second stage, cross-layer IoU trajectories, ground-truth box scale, and the winner query's localization advantage are used to impose budget-constrained adaptive margin ranking on persistent queries competing for the same target. Across the five-seed experiment, CQ-RT-DETR achieves a mean AP_{50–95} of 0.8892, improving by 2.01 percentage points over B_Lite, while reducing the parameter count and GFLOPs by 16.93% and 23.38%, respectively, relative to RT-DETRv2; its difference in accuracy from RT-DETRv2 is not significant. In the FP16 model-level test on the RK3576 platform, mean forward-pass latency decreases from 149.44 ms to 67.89 ms and FPS increases from 6.69 to 14.7. The principal advantage of CQ-RT-DETR is therefore its ability to recover the compression-induced accuracy loss without increasing inference-graph complexity and to provide a favorable accuracy–efficiency trade-off under the present dataset and hardware conditions.

Supplementary Materials: The following supporting information can be downloaded at <https://www.mdpi.com/article/10.3390/agronomy16181760/s1>. Table S1: Training configuration for different detection frameworks; Table S2: Published CottonWeedDet12 detection results (different experimental protocols; for background reference only); Table S3: Class-wise performance for the 12 weed classes (means across five seeds); Table S4: Standard deviations of class-wise performance for the 12 weed classes.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L.; software, Y.L.; validation, Y.L.; formal analysis, Y.L.; investigation, Y.L.; resources, L.W.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L.; visualization, Y.L.; supervision, L.W.; project administration, Y.L.; funding acquisition, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in CottonWeed-Det12 at <https://zenodo.org/records/7535814> (accessed on 18 May 2026). The source code and configuration files required to reproduce the proposed method will be made publicly available upon publication of this article at <https://github.com/BiggerBinBin/CQ-RT-DETR> (accessed on 1 August 2026).

Acknowledgments: The authors acknowledge the School of Information Science and Technology, Gansu Agricultural University, for providing computing resources and laboratory facilities.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RT-DETR	Real-Time Detection Transformer
SR-TCCM	Scale-Reliability-Guided Trajectory-Consistent Competition Margin

AP	Average Precision
AR	Average Recall
IoU	Intersection over Union
FPS	Frames Per Second
NPU	Neural Processing Unit

References

- Manalil, S.; Coast, O.; Werth, J.; Chauhan, B.S. Weed Management in Cotton (*Gossypium hirsutum* L.) through Weed-Crop Competition: A Review. *Crop Prot.* **2017**, *95*, 53–59. [\[CrossRef\]](#)
- Oerke, E.-C. Crop Losses to Pests. *J. Agric. Sci.* **2006**, *144*, 31–43. [\[CrossRef\]](#)
- Darbyshire, M.; Coutts, S.; Bosilj, P.; Sklar, E.; Parsons, S. Review of Weed Recognition: A Global Agriculture Perspective. *Comput. Electron. Agric.* **2024**, *227*, 109499. [\[CrossRef\]](#)
- Liu, B.; Bruch, R. Weed Detection for Selective Spraying: A Review. *Curr. Robot. Rep.* **2020**, *1*, 19–26. [\[CrossRef\]](#)
- Wu, Z.; Chen, Y.; Zhao, B.; Kang, X.; Ding, Y. Review of Weed Detection Methods Based on Computer Vision. *Sensors* **2021**, *21*, 3647. [\[CrossRef\]](#) [\[PubMed\]](#)
- Rahman, A.; Lu, Y.; Wang, H. Performance Evaluation of Deep Learning Object Detectors for Weed Detection for Cotton. *Smart Agric. Technol.* **2023**, *3*, 100126. [\[CrossRef\]](#)
- Hasan, A.S.M.M.; Sohel, F.; Diepeveen, D.; Laga, H.; Jones, M.G.K. A Survey of Deep Learning Techniques for Weed Detection from Images. *Comput. Electron. Agric.* **2021**, *184*, 106067. [\[CrossRef\]](#)
- Johnson, J.M.; Khoshgoftaar, T.M. Survey on Deep Learning with Class Imbalance. *J. Big Data* **2019**, *6*, 27. [\[CrossRef\]](#)
- Dyrmann, M.; Karstoft, H.; Midtby, H.S. Plant Species Classification Using Deep Convolutional Neural Network. *Biosyst. Eng.* **2016**, *151*, 72–80. [\[CrossRef\]](#)
- Milioto, A.; Lottes, P.; Stachniss, C. Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: Brisbane, QLD, Australia, 2018; pp. 2229–2235.
- Lottes, P.; Behley, J.; Milioto, A.; Stachniss, C. Fully Convolutional Networks With Sequential Information for Robust Crop and Weed Detection in Precision Farming. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2870–2877. [\[CrossRef\]](#)
- Olsen, A.; Konovalov, D.A.; Philippa, B.; Ridd, P.; Wood, J.C.; Johns, J.; Banks, W.; Girgenti, B.; Kenny, O.; Whinney, J.; et al. DeepWeeds: A Multiclass Weed Species Image Dataset for Deep Learning. *Sci. Rep.* **2019**, *9*, 2058. [\[CrossRef\]](#) [\[PubMed\]](#)
- Espejo-Garcia, B.; Mylonas, N.; Athanasakos, L.; Fountas, S.; Vasilakoglou, I. Towards Weeds Identification Assistance through Transfer Learning. *Comput. Electron. Agric.* **2020**, *171*, 105306. [\[CrossRef\]](#)
- Chen, D.; Lu, Y.; Li, Z.; Young, S. Performance Evaluation of Deep Transfer Learning on Multi-Class Identification of Common Weed Species in Cotton Production Systems. *Comput. Electron. Agric.* **2022**, *198*, 107091. [\[CrossRef\]](#)
- Dang, F.; Chen, D.; Lu, Y.; Li, Z. YOLOWeeds: A Novel Benchmark of YOLO Object Detectors for Multi-Class Weed Detection in Cotton Production Systems. *Comput. Electron. Agric.* **2023**, *205*, 107655. [\[CrossRef\]](#)
- Fan, X.; Sun, T.; Chai, X.; Zhou, J. YOLO-WDNet: A Lightweight and Accurate Model for Weeds Detection in Cotton Field. *Comput. Electron. Agric.* **2024**, *225*, 109317. [\[CrossRef\]](#)
- Zheng, L.; Long, L.; Zhu, C.; Jia, M.; Chen, P.; Tie, J. A Lightweight Cotton Field Weed Detection Model Enhanced with EfficientNet and Attention Mechanisms. *Agronomy* **2024**, *14*, 2649. [\[CrossRef\]](#)
- Hu, J.; Gong, H.; Li, S.; Mu, Y.; Guo, Y.; Sun, Y.; Hu, T.; Bao, Y. Cotton Weed-YOLO: A Lightweight and Highly Accurate Cotton Weed Identification Model for Precision Agriculture. *Agronomy* **2024**, *14*, 2911. [\[CrossRef\]](#)
- Lu, Z.; Chengao, Z.; Lu, L.; Yan, Y.; Jun, W.; Wei, X.; Ke, X.; Jun, T. Star-YOLO: A Lightweight and Efficient Model for Weed Detection in Cotton Fields Using Advanced YOLOv8 Improvements. *Comput. Electron. Agric.* **2025**, *235*, 110306. [\[CrossRef\]](#)
- Wang, S.; Wei, L. AVGS-YOLO: A Quad-Synergistic Lightweight Enhanced YOLOv11 Model for Accurate Cotton Weed Detection in Complex Field Environments. *Agriculture* **2026**, *16*, 828. [\[CrossRef\]](#)
- Li, A.; Zhang, K.; Wang, B.; Xu, C.; Liu, J. YOLO-CottWed: A Lightweight Network for Fine-Grained Weed Detection in Cotton Fields. *Neurocomputing* **2026**, *681*, 133404. [\[CrossRef\]](#)
- Allmendinger, A.; Saltık, A.O.; Peteinatos, G.G.; Stein, A.; Gerhards, R. Assessing the Capability of YOLO- and Transformer-Based Object Detectors for Real-Time Weed Detection. *Precis. Agric.* **2025**, *26*, 52. [\[CrossRef\]](#)
- Zhang, C.; Xiao, J.; Chang, Y. CEAM-DETR: An NMS-Free Lightweight Transformer for Weed Detection in Soybean Fields under Complex Conditions. *Sci. Rep.* **2026**, *16*, 23590. [\[CrossRef\]](#) [\[PubMed\]](#)
- Zhang, Y.; Xu, Y.; Ma, C.; Jiang, Y.; Song, Y. SCR-DETR: A Real-Time Lightweight DETR Model for Weed Detection. *J. Real-Time Image Proc.* **2025**, *22*, 126. [\[CrossRef\]](#)

25. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In *Computer Vision—ECCV 2020*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2020; Volume 12346, pp. 213–229.
26. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. *arXiv* **2020**. [[CrossRef](#)]
27. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs Beat YOLOs on Real-Time Object Detection. In *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Seattle, WA, USA, 2024; pp. 16965–16974.
28. Lv, W.; Zhao, Y.; Chang, Q.; Huang, K.; Wang, G.; Liu, Y. RT-DETRv2: Improved Baseline with Bag-of-Freebies for Real-Time Detection Transformer. *arXiv* **2024**, arXiv:2407.17140.
29. Wang, S.; Xia, C.; Lv, F.; Shi, Y. RT-DETRv3: Real-Time End-to-End Object Detection with Hierarchical Dense Positive Supervision. In *Proceedings of the 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*; IEEE: Tucson, AZ, USA, 2025; pp. 1628–1636.
30. Liu, S.; Ren, T.; Chen, J.; Zeng, Z.; Zhang, H.; Li, F.; Li, H.; Huang, J.; Su, H.; Zhu, J.; et al. Detection Transformer with Stable Matching. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV)*; IEEE: Paris, France, 2023; pp. 6468–6477.
31. Cai, Z.; Liu, S.; Wang, G.; Ge, Z.; Zhang, X.; Huang, D. Align-DETR: Enhancing End-to-End Object Detection with Aligned Loss. *arXiv* **2023**, arXiv:2304.07527.
32. Pu, Y.; Liang, W.; Hao, Y.; Yuan, Y.; Yang, Y.; Zhang, C.; Hu, H.; Huang, G. Rank-DETR for High Quality Object Detection. In *Proceedings of the Advances in Neural Information Processing Systems*; Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2023; Volume 36, pp. 16100–16113.
33. Zhang, H.; Wang, Y.; Dayoub, F.; Sunderhauf, N. VarifocalNet: An IoU-Aware Dense Object Detector. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Nashville, TN, USA, 2021; pp. 8510–8519.
34. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Computer Vision—ECCV 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2014; Volume 8693, pp. 740–755.
35. Microsoft. ONNX Runtime. Available online: <https://github.com/microsoft/onnxruntime> (accessed on 7 May 2026).
36. Varghese, R.; Sambath, M. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In *Proceedings of the 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*; IEEE: Chennai, India, 2024; pp. 1–6.
37. Khanam, R.; Hussain, M. YOLOv11: An Overview of the Key Architectural Enhancements. *arXiv* **2024**, arXiv:2410.17725.
38. Jocher, G.; Qiu, J.; Liu, M.; Lyu, S.; Akyon, F.C.; Kalfaoglu, M.E. Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. *arXiv* **2026**, arXiv:2606.03748.
39. Peng, Y.; Li, H.; Wu, P.; Zhang, Y.; Sun, X.; Wu, F. D-FINE: Redefine Regression Task in DETRs as Fine-Grained Distribution Refinement. *arXiv* **2024**, arXiv:2410.13842.
40. Zhou, Q.; Li, H.; Cai, Z.; Zhong, Y.; Zhong, F.; Lin, X.; Wang, L. YOLO-ACE: Enhancing YOLO with Augmented Contextual Efficiency for Precision Cotton Weed Detection. *Sensors* **2025**, *25*, 1635. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.