Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Compact Detection of Rebar Intersections and Tying States: Computational and Cross-Acquisition Trade-offs

Yuanxun Zheng
guo qiling
guoqiling86@gmail.com

Zhengzhou University https://orcid.org/0009-0009-8004-2481
Qiling Guo
Xiangyang Chen
Junwei Shi
Yanyun Li
Yihan Chen

Research Article

Keywords: rebar intersection, tying-state detection, compact object detector, YOLO11, construction automation

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11685053/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Compact Detection of Rebar Intersections and Tying States: Computational and Cross-Acquisition Trade-offs

Yuanxun Zheng[1]*, Qiling Guo[1], Xiangyang Chen[2], Junwei Shi[2], Yanyun Li[3], Yihan Chen[1]

[1] School of Water Resources and Transportation Engineering, Zhengzhou University, Zhengzhou 450001, China

[2] Zhengzhou Engineering Co., Ltd., China Railway Seventh Engineering Group, Zhengzhou, China

[3] Jianke Intelligent Equipment Manufacturing (Tianjin) Co., Ltd., Tianjin 300400, China

* Correspondence: yxzheng@zzu.edu.cn

Abstract – Automated rebar tying requires intersection locations and tying-state labels within a constrained computing budget. This study evaluates SATE-YOLO, a compact YOLO11n configuration, through component controls, paired training seeds, and measured device costs. Split-branch downsampling and a grouped prediction head reduce the model to 1.839 million parameters and 4.164 billion profiled floating-point operations at 640 × 640, reductions of 29.0% and 34.7%. Spatially weighted kernels admit an equivalent ordinary-convolution implementation. Two acquisition-day holdouts contain 375 and 159 annotated images. Across three seeds, mean average precision at an intersection-over-union threshold of 0.50 is 13.22% versus 10.02% for YOLO11n on the first holdout and 22.41% versus 24.78% on the second. At confidence 0.25, mean unbound recall is 3.44% and 32.70%, respectively. Component controls distinguish resource savings from recognition effects: removing attention gives 17.21% average precision in the first holdout with seed zero. On a GTX 1650 Ti, folded SATE-YOLO processes a decoded frame to host-memory detections with round medians of 15.44–15.55 ms and peak allocated tensor memory of 73.60 MiB. The evaluation identifies a compact operating point and shows why computational savings, acquisition transfer, and fixed-threshold target coverage require separate assessment for automated tying.

Keywords: rebar intersection; tying-state detection; compact object detector; YOLO11; construction automation

1 Introduction

In automated rebar tying, the next operation depends on both where an intersection lies and whether it has already been tied. Detecting intersections without state recognition can repeatedly select completed points. State recognition without localization cannot supply a target to the motion planner. A compact joint detector therefore provides a useful image-space interface: each visible intersection is assigned a location and a bound or unbound label.

Tying state is expressed through local wire shape and appearance around a bar crossing. Repeated steel bars, rust, shadows, and the texture beneath the mesh produce competing visual structure. At the same time, the detector must fit the computing resources available beside the tying equipment. Reducing operations indiscriminately can change the representation used for state discrimination. The engineering problem is to allocate computation across the network while preserving the evidence required by this two-class task.

Recent studies provide a direct foundation for this problem. Duan et al. developed YOLO-FAS for lightweight intersection and tying-state detection, including compression and robot deployment [1]. Xu et al. combined improved YOLOv5 with depth classification and thinning for binding-point localization [2]. Tan et al. integrated machine vision and coverage path planning in a rebar-tying robot [3]. These approaches address related but distinct outputs: joint detection and state recognition, geometric localization, and execution of a tying sequence. The comparison in this study is aligned with these task and sensor requirements.

SATE-YOLO uses a coordinated configuration of established components within YOLO11n. ADown changes the cost of resolution reduction; a grouped-convolution head provides a common feature stem for the two prediction tasks at each scale; cross-dimensional attention weights the transformed features; and a fixed spatial kernel template defines a controlled feature-extraction variant. The template can be folded into ordinary convolution weights at inference, preserving a conventional execution path.

The contribution is a controlled engineering assessment of where computation can be removed from this joint detector and how the resulting operating point transfers across acquisitions. The study separates three questions: which components supply resource savings, whether recognition changes persist over paired training seeds, and how the executable model behaves from a decoded frame to host-memory detections. A common two-class protocol includes a fully specified ShuffleNetV2–DualConv domain adaptation and a parameter-matched standard detector. Unbound precision and recall accompany aggregate AP because they determine the candidate targets available to downstream tying operations.

2 Related work

2.1 Rebar perception and tying systems

YOLO-FAS is the closest task-level comparator because it detects both intersection locations and tying states [1]. The method of Xu et al. combines detector proposals, depth-based separation, and thinning to estimate binding-point locations [2]. Tan et al. use RGB-D geometry and coverage planning within a tying robot [3]. FNB-YOLOv5 provides another detector for steel-mesh binding points, using FasterNet, BiFormer, a normalized Wasserstein distance loss, and an enhanced feature pyramid [4]. Li et al. connect recognition and localization with path planning [5].

More recently, Wang et al. studied a ShuffleNetV2 and DualConv-based YOLOv8 framework for joint intersection detection and segmentation [6]. This extends the task comparison to lightweight perception with contour outputs, while the present evaluation centers on compact two-class bounding-box detection.

Recent work also extends perception to keypoint-guided localization. Dong et al. combine YOLOv8n-pose-Binding with target coordinate-frame construction and manipulator path planning [7]. These pose outputs use keypoint annotations in addition to the intersection boxes and tying-state labels considered here.

Wang et al. use geometry-only synthetic examples to generate pseudo-labels for real rebar images and then train a detector for tying across joint topologies [8]. Their perception-to-action pipeline addresses joint geometry and collision-free pose generation. The present study measures RGB box localization and tying-state recognition across acquisition days.

Table 1 Related rebar methods and their comparison roles

Study	Task or main approach	Comparison role
Duan et al. [1]	Lightweight intersection and tying-state detection	Closest task-level reference
Xu et al. [2]	Improved YOLOv5, depth classification, thinning	Localization with the required depth input
Tan et al. [3]	RGB-D perception and coverage planning	System and geometric localization context
Li et al. [4]	FNB-YOLOv5 binding-point detection	Additional specialist detector
Li et al. [5]	Recognition, localization, path planning	Perception-to-motion interface

Table 1 distinguishes bounding-box state detection from depth-based localization and robotic execution. Same-split detector evaluation supports a comparison of recognition quality and computational cost. Spatial error and completed tying success require their corresponding calibration, sensors, and execution measurements; published scores from different datasets are used as context rather than combined into a numerical ranking.

2.2 Compact detection architectures

YOLO11 supplies the multi-scale detection architecture [9]. ADown, available in the YOLOv9 implementation, combines pooled feature aggregation with channel splitting and two downsampling branches [10]. Triplet Attention models interactions between channel and spatial axes [11]. These mechanisms act at different stages of feature processing. SATE-YOLO examines their placement together with a grouped prediction stem, with component-removal and kernel-matched controls used to identify the source of the measured operating trade-off.

3 Detection architecture

3.1 Joint intersection and state detection

For an RGB image, the network outputs bounding boxes, confidence scores, and one of two labels: bound or unbound. A box encloses an intersection and the local appearance used to identify its state. Its center provides an image-coordinate target for downstream processing. The backbone and feature pyramid preserve predictions at strides 8, 16, and 32.

Figure 1 specifies the complete two-branch configuration. Weighted C3k2 blocks appear at layers 2, 4, 6, 8, 14, 17, 20, and 23. ADown replaces resolution-reduction stages at layers 3, 5, 7, 18, and 21. Attention follows C2PSA at layer 11. The head receives layers 17, 20, and 23; the connection from layer 10 to layer 22 supplies features directly to the deepest prediction path.

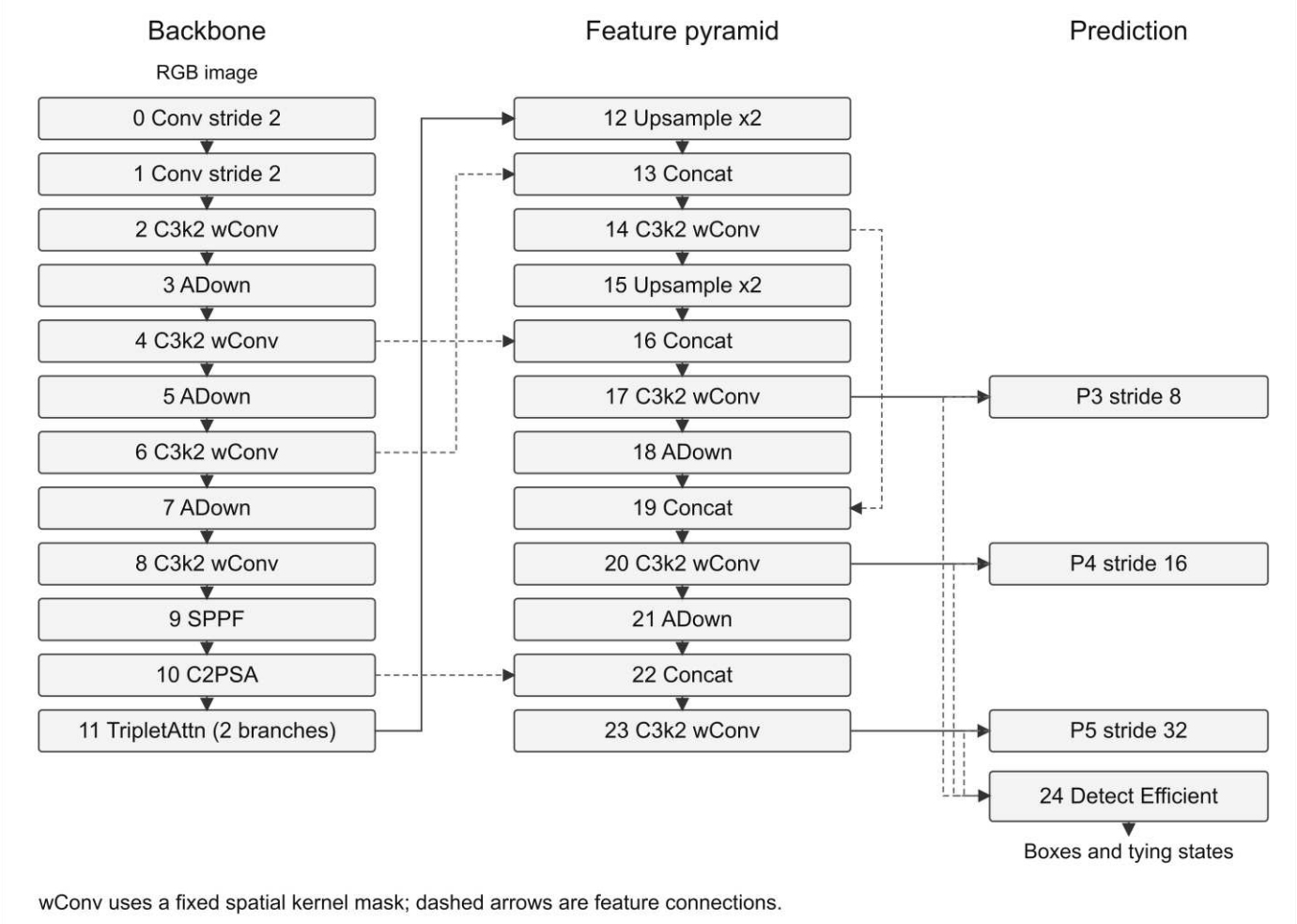

Fig. 1 Layer specification of the two-branch SATE-YOLO configuration

3.2 Spatially weighted feature extraction

The weighted bottleneck adopts the spatial-density formulation of Cammarasana and Patanè [12], using a learned 3 × 3 kernel K and a fixed template Φ. The template in Equation (1) has center value 1.0, axial values 0.9, and corner values 0.81.

$$\Phi = \begin{bmatrix} 0.81 & 0.90 & 0.81 \\ 0.90 & 1.00 & 0.90 \\ 0.81 & 0.90 & 0.81 \end{bmatrix} \quad (1)$$

The effective kernel is the elementwise product in Equation (2), with the template broadcast across the input and output channel axes.

$$K_{\text{eff}} = K \odot \Phi \quad (2)$$

Both internal convolutions in the selected bottleneck use this operator. The surrounding C3k2 split, concatenation, and projection operations are preserved. The coefficient 0.9 is fixed before the revised experiments.

All entries of Φ are positive, so each dense effective kernel corresponds to exactly one raw kernel K. Under a plain gradient step on K, the step in effective-kernel coordinates is the gradient with respect to the effective kernel multiplied elementwise by Φ squared. The resulting learning-rate multipliers are 1.00 at the center, 0.81 at axial positions, and 0.6561 at corners. This calculation describes the data-loss update without momentum or weight decay. The trained all-ones control tests the combined effect of the template on initialization and optimization.

The implementation applies the weighted convolutions directly, while the surrounding projection layers retain their usual normalization and activation. Consequently, the kernel-template control uses the same bottleneck with Φ set to an all-ones matrix. This holds normalization and activation placement fixed while isolating the effect of spatial weighting. The separate component-removal control restores the standard C3k2 block.

At inference, K_eff is computed once and stored as the weight of an ordinary convolution. The resulting layer has the same stride, padding, dilation, and groups. This transformation removes runtime template multiplication while preserving the convolution output. Computational reductions in the complete network are measured from its architecture rather than attributed to the nonzero template alone.

3.3 Split-branch downsampling

ADown first applies stride-one average pooling and splits the channels into two groups [10]. One branch uses a stride-two 3 × 3 convolution. The other uses stride-two max pooling followed by a 1 × 1 convolution. Their concatenation produces the downsampled output. The two branches retain different local aggregation paths while controlling the cost of spatial reduction.

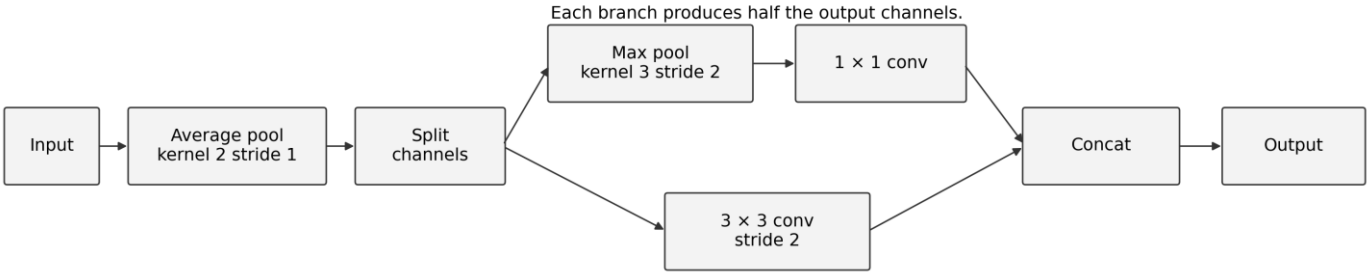

Fig. 2 ADown resolution-reduction block

3.4 Cross-dimensional attention

The main configuration uses the two cross-dimensional branches of the Triplet Attention implementation [11]. One branch permutes the channel and height axes and the other permutes the channel and width axes. Each branch applies ZPool, a learned 7 × 7 convolution with batch normalization, and a sigmoid gate. The weighted tensors are restored to the original axes and averaged. The two-branch module contains 200 trainable parameters.

A separate three-branch configuration adds the direct spatial branch and averages all three outputs. It contains 300 attention parameters and is trained independently. The two- and three-branch settings are explicit Boolean options in the executable configuration, allowing their operating trade-offs to be compared without changing trained checkpoints.

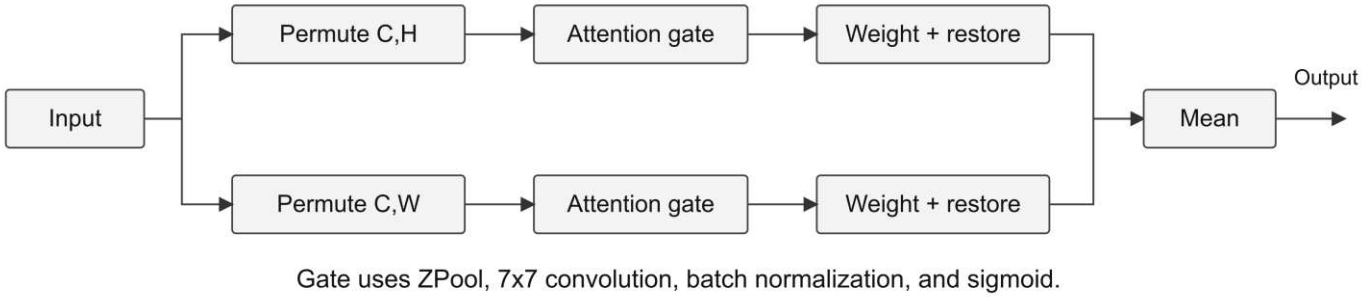

Fig. 3 Two-branch cross-dimensional attention

3.5 Grouped prediction head

At each prediction scale, the adopted Detect_Efficient head [13] applies two 3 × 3 grouped convolutions, each followed by batch normalization and SiLU. For C input channels, the group count is C/16. The three input channel counts are 64, 128, and 256. Each scale has its own feature stem. Within that scale, the resulting features are shared by classification and box regression.

Separate 1 × 1 projections produce two class logits and 4 × 16 box-regression values. Distribution-based decoding converts the regression outputs to box coordinates. Figure 4 shows the scale-local sharing arrangement. The head thus avoids duplicating the intermediate task stems while retaining separate output projections.

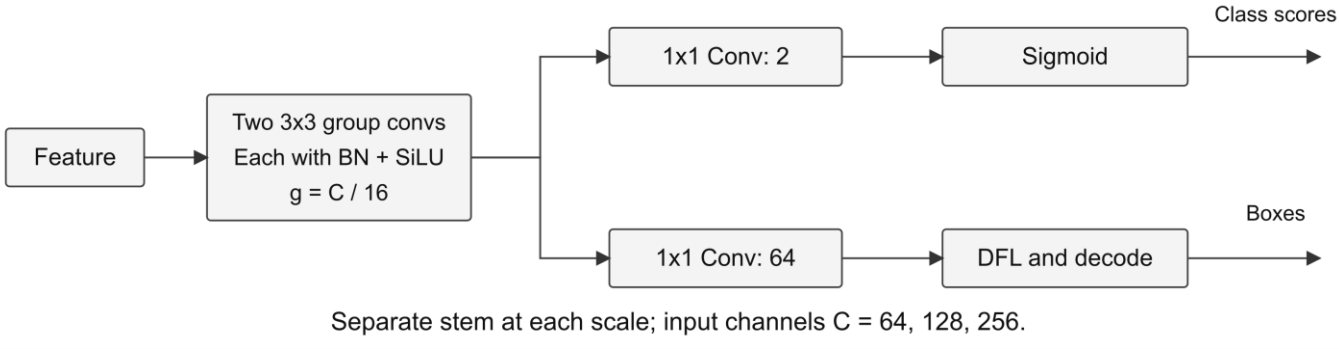

Fig. 4 Grouped feature stem and task-specific outputs at each scale

The grouped-convolution option is adopted from the YOLO11 modification project documented in [13]. The local implementation and its configuration are archived with the experiments.

4 EXPERIMENTAL PROTOCOL

4.1 Image pools and acquisition grouping

The annotation task uses two classes, bound and unbound. The available legacy pool contains 830 files without the identified augmentation suffixes. A later acquisition contains 550 images with EXIF capture dates of 16–17 December 2025; 534 have valid existing two-class annotations. The other 16 are excluded for missing annotation, using a rule frozen before inference. The later pool contains 6,644 annotated intersections: 3,345 bound and 3,299 unbound. Table 2 summarizes these image pools.

The legacy records contain 6,483 annotations, with 3,286 bound and 3,197 unbound targets. Original resolutions vary, and most later records are 3456 × 3456 pixels. Available EXIF metadata identifies vivo iQOO Neo7 SE and Xiaomi devices in the acquisition pools. Images retain their recorded aspect ratio during resizing and letterboxing to the detector input; normalized class and box annotations are transformed with the image.

Table 2 Image pools for the revised evaluation protocol

Pool	Image records	Use in the protocol
Legacy archive	830	Retained together in the training pool
16 December 2025	159 annotated; 2 missing labels	Capture-day holdout or training complement
17 December 2025	375 annotated; 14 missing labels	Capture-day holdout or training complement

The legacy pool is treated as a single acquisition group because scene matching identified overlapping mesh regions in its former file-level splits. Augmentation derivatives are excluded from the validation and test lists. The revised capture-day protocol holds out one later day at a time: the 17 December fold has 989 training images and 375 test images; the 16 December fold has 1,205 training images and 159 test images. Split manifests contain file paths, annotation paths, acquisition groups, and hashes.

Capture-day grouping defines the scope of this evaluation. A different date is not a physical-mesh identity, and multiple views of one mesh are not independent meshes. The reported outcomes therefore characterize transfer across the declared acquisition groups. Physical-mesh generalization requires identity records or separately acquired meshes.

A candidate scene-matching audit tested 250 retrieved pairs between the legacy and later pools and 250 pairs between the later capture days. No tested pair met the high-match criterion of at least 30 homography inliers and 4% image coverage. Visual review of the six highest-ranked pairs in each category found no confirmed overlapping region. This is a retrieval audit rather than proof of independent physical-mesh identities.

Three annotators prepared the source-image annotations using LabelImg, and two reviewers checked the annotations. The revised experiments retain the existing two-class box labels.

When wire appearance was unclear or an intersection was occluded, the tying-state label was determined during review.

4.2 Controlled training

The revised protocol fixes the training budget before the new runs. All configurations use 640 × 640 input, 80 epochs, SGD with an initial learning rate of 0.01, a nominal batch size of 64 through gradient accumulation, and FP32 execution. Training begins from random initialization. Seeds 0, 1, and 2 are predeclared for the principal comparisons. The selected checkpoint is the final epoch, so the held-out acquisition is not used to choose training duration or model weights. Table 3 lists the fixed training settings.

Table 3 Fixed settings for the revised training runs

Setting	Value
Input / epochs	640 × 640 / 80
Physical batch / nominal batch	4 / 64
Optimizer / initial learning rate	SGD / 0.01
Precision / initialization	FP32 / from scratch
Random seeds	0, 1, 2
Checkpoint selection	Epoch 80, last.pt
Held-out acquisition	Fixed manifest; no test-based tuning

The first controlled runs use a GTX 1650 Ti with 4 GB memory on an Intel Core i7-10750H Windows host, Python

3.9.21, PyTorch 2.0.0+cu117, and Ultralytics 8.3.141. SGD momentum is 0.937 and weight decay is 0.0005. The learning rate decreases linearly to 1% of its initial value, following a three-epoch warm-up. Augmentation uses translation 0.1, scale 0.5, horizontal flip probability 0.5, mosaic probability 1.0, and HSV gains of 0.015, 0.7, and 0.4. Rotation, shear, perspective, vertical flip, mixup, and copy-paste are disabled. Complete run arguments and split-manifest hashes are archived with the checkpoints.

4.3 Detection and resource measurements

AP50 is the average precision at an intersection-over-union threshold of 0.50. AP50–95 averages over thresholds from 0.50 to 0.95 in increments of 0.05. Mean AP averages the two classes. Aggregate and per-class results are reported together, with particular attention to unbound-point recall. AP integration uses confidence 0.001, class-aware nonmaximum suppression (NMS) IoU 0.7, and at most 300 detections per image. Evaluation uses batch one with the longest image side resized to 640 pixels and stride-aligned rectangular letterboxing. Operational precision and recall are recomputed at the predeclared confidence threshold of 0.25, with one-to-one class-consistent matching at IoU 0.50.

Parameter counts use the unfused model. Operations are profiled at 640 × 640 with PyTorch operator profiling, including functional convolutions within the weighted bottlenecks; a multiply–accumulate contributes two floating-point operations where supported by the profiler. The recorded operators include convolution, batch matrix multiplication, and supported elementwise additions and multiplications. Profiling does not fully account for normalization, pooling, activation, or data movement. Serialized bytes and peak runtime memory are treated as separate quantities.

Latency measurement uses batch one and records the platform, numerical precision, execution backend, warm-up count, and repeated measurement count. GPU work is synchronized at the timing boundaries. Model-forward latency and image-to-output latency are reported separately; the latter includes the declared preprocessing, data-transfer, inference, and postprocessing stages. Median and 95th-percentile latency describe both typical and delayed responses.

4.4 Comparators and component controls

YOLO11n is the common architecture baseline. YOLO-FAS [1] and FNB-YOLOv5 [4] establish the specialist task context; their published scores use different datasets and are not combined with the same-split numerical comparisons. The implemented domain control is the explicitly specified adaptation below. The depth-dependent methods in [2,3] are compared within their sensor and localization scope.

An implemented domain control follows the ShuffleNetV2 and DualConv design of Wang et al. [6]. The local adaptation uses the torchvision ShuffleNetV2 ×1.0 stages with 116, 232, and 464 output channels. Its YOLOv8 feature pyramid produces 64, 128, and 256 channels, using one bottleneck in each C2f block. Both internal 3 × 3 convolutions use the sum of a four-group spatial convolution and a full 1 × 1 projection, followed by batch normalization and SiLU. The detection outputs remain at strides 8, 16, and 32. This explicitly configured adaptation is trained from scratch under the same data, augmentation, and optimizer protocol.

The component analysis includes the complete two-branch configuration, the three-branch variant, four models removing W, A, T, or E, and the all-ones kernel-template control. Here W denotes the weighted bottleneck, A denotes ADown, T denotes attention, and E denotes the grouped head. YOLO11n and the predeclared two-branch model are trained with paired seeds 0, 1, and 2 in both acquisition folds. Means, sample standard deviations, and within-seed differences describe training variation. The domain adaptation, matched-width model, and component controls use seed zero; these are reported separately from the three-seed estimates.

The parameter-matched control retains the standard YOLO11 operators and changes its width multiplier to 26/128. It has 1,813,650 parameters, 1.37% fewer than the complete two-branch model. This width is selected by parameter proximity from the fixed grid 23/128 to 30/128, without using detection scores. Its profiled workload is 4.876 GFLOPs.

All 21 runs complete the fixed 80-epoch budget and use the last checkpoint. Three trajectories resume from saved checkpoints: the matched-width model after epoch 69, the 16 December two-branch seed-zero run after epoch 10, and the 16 December YOLO11n seed-two run after epoch 79. The framework restores EMA weights and optimizer state but does not archive random-number or data-loader states. Online Resource 1 identifies every trajectory, source configuration, split hash, and checkpoint hash.

4.5 AI assistance during revision

During manuscript revision, OpenAI Codex assisted with literature retrieval, preparation of audit and experimental scripts, experimental orchestration, result tabulation, and English drafting. The reported numerical measurements are generated by execution of the archived scripts on the stated platform. The authors retain responsibility for verification of the source data, scientific judgments, code, and final manuscript.

5 RESULTS

5.1 Computational budget

Table 4 summarizes the resource measurements. The complete two-branch configuration reduces the parameter count from 2,590,230 to 1,838,830, a reduction of 29.0%. The profiled workload decreases from 6.375 to 4.164 GFLOPs, a reduction of 34.7%. These measurements establish the network's resource budget independently of its detection scores. The three-branch configuration has 1,838,930 parameters, isolating the additional attention branch with a difference of 100 parameters.

Kernel folding was checked across all 22 weighted convolution modules. Three FP32 input trials produced identical decoded outputs before and after replacement with

ordinary convolutions, with a maximum absolute difference of 0.0 in the tested runtime. The parameter count remains 1,838,830. This establishes an equivalent inference implementation of the fixed template.

Table 4 Verified architecture resource measurements at 640 × 640

Configuration	Parameters (M)	Profiled GFLOPs
YOLO11n	2.590	6.375
SATE-YOLO, two branches	1.839	4.164
SATE-YOLO without ADown	2.318	5.186
SATE-YOLO with standard head	2.109	5.355
Standard YOLO11, matched parameters	1.814	4.876
ShuffleNetV2–DualConv adaptation	2.583	6.657

5.2 Cross-acquisition detection over paired seeds

Table 5 reports the two primary detectors over the same three training seeds in each acquisition direction. The 17 December holdout contains 375 images and 3,992 intersections, with 989 images used for training. The reverse fold contains 159 test images and 2,652 intersections, with 1,205 training images. Means and sample standard deviations describe seed variation within each fixed fold.

Table 5 Primary-model results, mean ± sample standard deviation over seeds 0, 1, and 2; all values in percent

Holdout	Model	mAP50	mAP50–95	Unbound P	Unbound R
17 Dec	YOLO11n	10.02 ± 0.66	3.62 ± 0.35	10.96 ± 2.80	5.97 ± 1.75
17 Dec	SATE-YOLO, two branches	13.22 ± 2.07	4.77 ± 0.91	7.30 ± 1.97	3.44 ± 1.13
16 Dec	YOLO11n	24.78 ± 1.52	7.66 ± 0.42	41.50 ± 3.45	36.13 ± 1.99
16 Dec	SATE-YOLO, two branches	22.41 ± 3.10	6.99 ± 1.02	40.93 ± 3.60	32.70 ± 6.90

With 17 December held out, the paired mAP50 differences (SATE-YOLO minus YOLO11n) are +2.06, +5.23, +2.33 percentage points for seeds 0, 1, and 2. Their mean is +3.21 points, with a sample standard deviation of 1.75 points.

With 16 December held out, the paired mAP50 differences (SATE-YOLO minus YOLO11n) are -2.16, +1.96, -6.89 percentage points for seeds 0, 1, and 2. Their mean is -2.37 points, with a sample standard deviation of 4.43 points.

The 17 December AP gain occurs in all three paired seeds. At the fixed confidence threshold of 0.25, however, mean unbound recall is 3.44% for SATE-YOLO and 5.97% for YOLO11n. The reverse fold gives 32.70% and 36.13%, respectively. AP integrates detection ranking across confidence thresholds, whereas these recall values describe retained unbound targets at a particular operating point. The reported AP changes therefore do not establish increased target coverage at that threshold. Per-class AP, individual seed results, paired differences, and resumed trajectories appear in Online Resource 1.

5.3 Same-split domain and capacity controls

Table 6 separates the seed-zero domain and architecture controls from the repeated primary comparison. Every row uses the same frozen fold, final epoch, two-class labels, and evaluation settings as its corresponding primary run. The ShuffleNetV2–DualConv row denotes the stated local adaptation of [6].

Table 6 Seed-zero detection controls by held-out acquisition day

Holdout	Model	mAP50 (%)	mAP50–95 (%)	Bound AP50 (%)	Unbound AP50 (%)
17 Dec	YOLO11n	10.46	3.88	8.27	12.65
17 Dec	SATE-YOLO, two branches	12.52	4.51	12.94	12.11
17 Dec	SATE-YOLO, three branches	14.84	5.50	14.84	14.85
17 Dec	Matched-width YOLO11	10.65	3.86	11.24	10.07
17 Dec	ShuffleNetV2–DualConv adaptation	6.54	2.27	7.40	5.69
16 Dec	YOLO11n	25.42	7.91	22.54	28.29
16 Dec	SATE-YOLO, two branches	23.25	7.33	20.92	25.59
16 Dec	ShuffleNetV2–DualConv adaptation	25.44	7.75	25.28	25.59

The domain adaptation obtains 6.54% mAP50 on the 17 December holdout and 25.44% on the reverse holdout. SATE-YOLO uses 28.8% fewer parameters and 37.5% fewer profiled operations than this adaptation. The standard matched-width control gives 10.65% mAP50 on 17 December at 1.814 million parameters and 4.876 GFLOPs, compared with 12.52% at 1.839 million parameters and 4.164 GFLOPs for SATE-YOLO. These seed-zero comparisons distinguish architecture allocation from the effect of reducing width alone.

Adding the direct spatial attention branch gives 14.84% mAP50, 14.85% unbound AP50, and 5.11% fixed-threshold unbound recall on 17 December. The two-branch primary configuration remains the same throughout the repeated-seed comparison; the exploratory variants do not replace it after inspecting held-out results.

5.4 Component trade-offs

Table 7 Component controls on the 17 December holdout with seed zero

Configuration	mAP50 (%)	mAP50–95 (%)	Parameters (M)	Profiled GFLOPs
Complete two-branch model	12.52	4.51	1.839	4.164
All-ones kernel template	7.01	2.30	1.839	4.164
Restore standard downsampling	10.43	3.82	2.318	5.186
Restore standard head	13.56	5.36	2.109	5.355
Restore standard C3k2	12.54	4.78	1.841	4.163
Remove attention	17.21	7.03	1.839	4.161

Table 7 identifies the resource allocation within the compact configuration. Restoring standard downsampling adds 479,232 parameters and 1.022 GFLOPs; restoring the standard detection head adds 270,464 parameters and 1.192 GFLOPs. ADown and the grouped head provide the principal architectural savings. The standard head gives 13.56% mAP50 versus 12.52% for the grouped head, establishing a recognition-cost trade-off in this seed-zero comparison.

The all-ones template gives 7.01% mAP50 at the same parameter and operation geometry as the weighted model. This control holds normalization and activation placement fixed and measures the template's combined initialization and optimization effect. Restoring standard C3k2 gives 12.54% mAP50 at nearly the same cost as the complete model. That comparison changes normalization and activation together with the convolution parameterization; the all-ones result consequently does not establish an advantage over the standard block.

Removing attention gives 17.21% mAP50 and 7.03% mAP50–95, with unbound AP50 of 18.47%. Fixed-threshold unbound precision and recall are 17.43% and 6.33%. This control removes 200 parameters while retaining the downsampling, weighted blocks, and grouped head. The simpler feature path is the favorable seed-zero accuracy choice in this fold. The component results support selecting modules by their measured resource and recognition roles rather than assuming that every added module improves both.

5.5 Device latency, memory and application output

Table 8 reports serial batch-one measurements on the GTX 1650 Ti using unfused PyTorch eager FP32 execution and 640 × 640 input. Twenty calls warm up each measurement boundary. Each of three rounds records 100 model-forward calls and 90 decoded-frame-to-output calls. Table entries span the three round-level medians or 95th percentiles; these ranges are not confidence intervals. Forward-only measurements, raw timing samples, and the 30/60/90 Hz sensitivity counts are provided in Online Resource 1.

Table 8 Pipeline latency and peak allocated CUDA tensor memory; latency ranges span three measurement rounds

Configuration	Pipeline P50 (ms)	Pipeline P95 (ms)	Peak (MiB)	Above 16.67 ms (%)
YOLO11n	17.11–19.93	20.33–32.67	81.99	71.11–87.78
Matched-width YOLO11	16.80–17.68	19.79–24.96	75.26	53.33–76.67
ShuffleNetV2–DualConv	18.55–19.84	23.43–24.59	103.88	98.89–100.00
SATE-YOLO	16.39–16.80	18.66–20.21	73.62	37.78–53.33
SATE-YOLO, folded	15.44–15.55	18.39–18.65	73.60	12.22–21.11
Three branches, folded	16.57–18.18	19.71–22.04	73.60	47.78–86.67
Without attention	16.91–17.35	19.03–20.04	78.65	63.33–77.78
Without attention, folded	16.81–17.06	19.25–22.84	73.59	58.89–65.56

The pipeline starts with a decoded blue-green-red (BGR) frame in host memory and ends with detections in native image coordinates in host memory. It includes letterboxing, tensor conversion, host-to-device (H2D) transfer, prediction and decoding, NMS, coordinate scaling, and device-to-host (D2H) transfer. Nine images occupy equally spaced indices in the frozen test list. File input, JPEG decoding, acquisition, and robot actuation lie outside the boundary. Peak memory measures PyTorch-allocated model and transient CUDA tensors, rather than total device memory.

Folded SATE-YOLO has round medians of 15.44–15.55 ms, compared with 17.11–19.93 ms for YOLO11n, and peak tensor allocations of 73.60 and 81.99 MiB. Relative to the unfolded SATE model, folding removes the explicit kernel-template multiplication and changes the median range from 16.39–16.80 ms to 15.44–15.55 ms. The attention-free folded control has medians of 16.81–17.06 ms despite its smaller architecture. Pipeline time depends on the executable network and its postprocessing outputs, as well as the profiled operation count.

The 60 Hz sensitivity period is 16.67 ms. Folded SATE-YOLO exceeds this period in 12.22–21.11% of the 90 calls per round; its 95th-percentile latency is 18.39–18.65 ms. A median below one frame period therefore does not guarantee that every frame is processed within it. These are empirical processing-period scenarios, not a measured robot deadline or a worst-case execution-time bound. The attention-free timing control was added after its seed-zero ablation result and supplies a runtime comparison, not an independent accuracy validation.

Figure 5 shows seed-zero outputs on the lower-median image of each frozen test list. Identical 1024-pixel crops center on the annotated intersection nearest each image center. Predictions come from full-image evaluation at confidence 0.25. YOLO11n returns a bound box in the 16 December crop; the two-branch model returns no retained box in either crop. The fixed examples illustrate localization and missed targets without defining an additional accuracy metric.

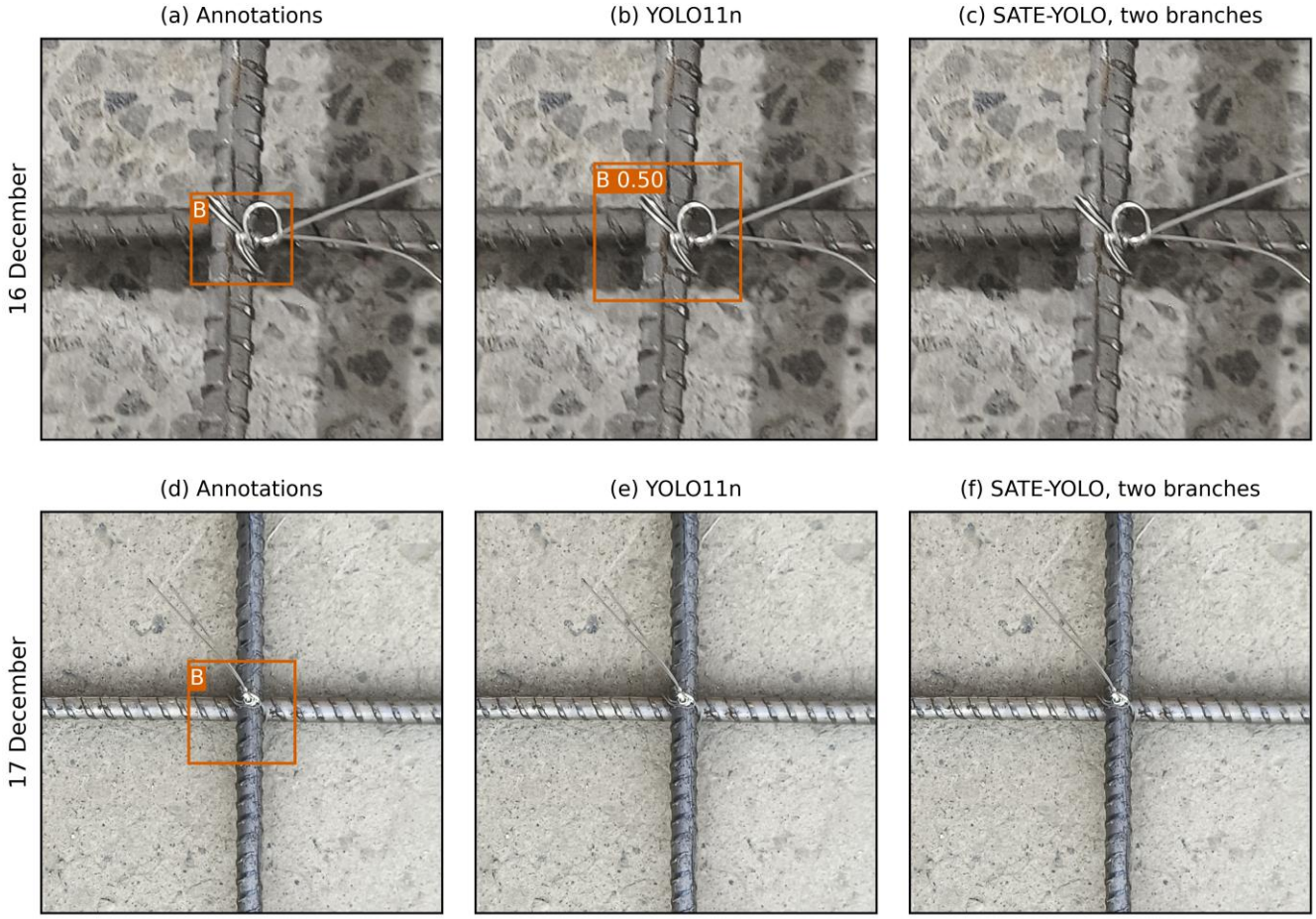

Fig. 5 Recorded annotations and seed-zero predictions on fixed holdout crops. Panels a–c use 16 December and d–f use 17 December. B denotes bound; prediction labels include confidence

6 Discussion

For rebar assembly perception, compactness is an allocation decision across feature extraction, resolution reduction, and prediction. The component controls locate the dominant savings in ADown and the grouped head. Kernel folding supplies an equivalent conventional-convolution execution path, while the positive kernel template changes training parameterization without reducing dense kernel capacity. These distinctions identify which design decisions reduce architecture cost and which require recognition evidence of their own.

The repeated-seed results establish acquisition-dependent operating trade-offs. The paired mean mAP50 change is +3.21 percentage points on 17 December and −2.37 points on 16 December; sample standard deviations of the paired changes are 1.75 and 4.43 points. These values describe the two declared folds, with resumed trajectories explicitly identified. The existing archive has no complete physical-mesh identity record, so seed variation and capture dates do not establish independent-mesh uncertainty. Image scale, scene appearance, and training composition change together across these folds and are not isolated causal factors.

The output available to a tying controller is a set of image-coordinate boxes and bound/unbound labels. AP and resource savings characterize the detector, while the fixed-threshold precision and recall characterize the candidate set for that controller. The observed target coverage does not establish unattended tying performance. Calibrated spatial localization and completed tying operations require corresponding system measurements. The measured GPU pipeline and its period-exceedance rates help select a perception implementation without converting typical latency into a robot-cycle guarantee.

7 Conclusion

This study provides a compact rebar intersection and tying-state configuration together with a controlled assessment of its computational and recognition trade-offs. The model reduces parameters by 29.0% and profiled operations by 34.7% relative to YOLO11n. ADown and the grouped prediction head supply the principal savings, and the weighted kernels admit an equivalent folded implementation. Across paired seeds, acquisition-day mAP50 changes are +3.21 and −2.37 percentage points, while the component controls favor the attention-free path in the tested seed-zero fold. On the GTX 1650 Ti, folded SATE-YOLO yields decoded-frame-to-output round medians of 15.44–15.55 ms with 73.60 MiB peak allocated tensor memory. The resulting evidence separates architecture cost, executable latency, and unbound-target

coverage for selecting perception models in automated rebar assembly.

Supplementary information

Online Resource 1: Training and acquisition protocols, all 21 per-run results, paired-seed summaries, checkpoint and implementation hashes, and the eight-configuration device timing and memory records. Online Resource 2: Raw model-forward and decoded-frame-to-output timing samples with configuration, round, and sample identifiers.

Statements and Declarations

Funding

This work was supported by the Henan Provincial Science and Technology R&D Plan Joint Fund under Grant No. 235200810050.

Data availability

The images and annotations, trained model weights, and experimental scripts developed for this study are available from the corresponding author at yxzheng@zzu.edu.cn upon reasonable request. Third-party software is identified by its cited sources.

Ethics and consent

The study does not involve human participants or animal experiments. Ethics approval, informed consent, and consent to publish are not applicable.

Author contributions

Yuanxun Zheng: Research design and experimental planning and execution. Qiling Guo: Data processing and analysis. Xiangyang Chen: Visualization.

Junwei Shi: Interpretation of results. Yanyun Li: Figure preparation. Yihan Chen: Literature search. All authors approved the revised manuscript and its submission.

Competing interests

The authors declare no competing financial or non-financial interests related to this work.

References

[1] Duan H, Yu M, Ai T, Zhu M, Jiang H, Guo S. YOLO-FAS: A lightweight model for detecting rebar intersections location and tying status. Neurocomputing. 2025;624:129485. https://doi.org/10.1016/j.neucom.2025.129485

[2] Xu K, Lu X, Shen T, Zhu X, Wang S, Wang X, Wang J. Rebar binding point location method based on improved YOLOv5 and thinning algorithm. Measurement. 2025;242:116029. https://doi.org/10.1016/j.measurement.2024.116029

[3] Tan X, Xiong L, Zhang W, Zuo Z, He X, Xu Y, Li F. Rebar-tying Robot based on machine vision and coverage path planning. Robotics and Autonomous Systems. 2024;182:104826. https://doi.org/10.1016/j.robot.2024.104826

[4] Li Z, Zhao Z, Zhang Z, Jie J, Cheng R. Object Detection of Steel Mesh Binding Point Using FNB-YOLOv5. Journal of Shanghai Jiao Tong University. 2026;60(5):762–775. https://doi.org/10.16183/j.cnki.jsjtu.2024.121

[5] Li C, Zhang W, Li F, et al. Vision-Based Detection, Localization, and Optimized Path Planning for Rebar Intersections in Automated Construction. Applied Sciences. 2025;15(13):7186. https://doi.org/10.3390/app15137186

[6] Wang R, Shi F, She Y, Zhang L, Lin K, Fu L, Shi J. A Lightweight Improved YOLOv8-Based Method for Rebar Intersection Detection. Applied Sciences. 2025;15(24):12898. https://doi.org/10.3390/app152412898

[7] Dong L, Zhang R, Shao Z, Bian Z, Wang X. Binding Point Recognition and Localization and Manipulator Binding Path Planning for a Rebar Binding Robot. Sensors. 2026;26(4):1315. https://doi.org/10.3390/s26041315

[8] Wang H, Jin Z, Shao Y. Geometry-driven perception and action for topology-agnostic robotic rebar tying. Automation in Construction. 2026;188:107013. https://doi.org/10.1016/j.autcon.2026.107013

[9] Ultralytics. Ultralytics YOLO11 documentation. https://docs.ultralytics.com/models/yolo11/ (accessed 30 October 2026).

[10] Wang C Y, Yeh I H, Liao H Y M. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. European Conference on Computer Vision. 2024. Official implementation: https://github.com/WongKinYiu/yolov9

[11] Misra D, Nalamada T, Arasanipalai A U, Hou Q. Rotate to Attend: Convolutional Triplet Attention Module. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021:3139–3148.

[12] Cammarasana S, Patanè G. Optimal Weighted Convolution for Classification and Denosing. arXiv preprint. 2025;2505.24558. https://doi.org/10.48550/arXiv.2505.24558. Author implementation: https://github.com/cammarasana123/weightedConvolution2.0

[13] z1069614715. YOLO11 and YOLO12 improvement project. objectdetection_script, public project documentation. https://github.com/z1069614715/objectdetection_script/blob/master/yolo-improve/yolov11-project.md (accessed 31 October 2026).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- ESM1.pdf
- ESM2.csv