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.

Accuracy and CPU speed gains across lightweight YOLO generations under downstream uncertainty

Ge Yang

1502090@peizheng.edu.cn

Guangdong Peizheng College

Article

Keywords:

Posted Date: November 1st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11226885/v1

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

Additional Declarations: No competing interests reported.

Abstract

Each new lightweight YOLO release reports higher COCO accuracy and lower CPU latency, and applied studies commonly build on the newest release. Recent multi-dataset comparisons show that a newer release need not win downstream, but they rarely separate a model difference from training-seed and test-set variability. We trained six nano-scale generations with three seeds on floating-debris and PASCAL VOC subsamples and three generations on VisDrone (49 runs), and compared them with paired test-image resampling. YOLO26n led YOLOv8n by 5.98 percentage points of mAP@0.5:0.95 on VOC but trailed it by 1.62 points on floating debris and 0.69 points on VisDrone, with all three bootstrap intervals excluding zero. The floating-debris deficit occurred in every object-size stratum and persisted under the default training recipe. These pairwise contrasts were stable, whereas the complete six-model ranking survived in only 55.1% and 16.4% of resamples. On three CPUs, randomized interleaved timing gave latency reductions of 1–12% from YOLOv8n to YOLO11n and 14–22% from YOLO11n to YOLO26n, against about 30% and 31% implied by the published tables. A release-level gain is therefore a hypothesis to test on the target data and processor. Pairwise intervals, not full rankings, should decide an upgrade or a baseline.

Introduction

Lightweight one-stage detectors of the YOLO (You Only Look Once) family are widely used in applied object detection. The Ultralytics framework provides training, evaluation and export interfaces for several generations [1–5]. Release documentation reports accuracy on Common Objects in Context (COCO) [6] and, for some models, CPU latency. The nano-scale COCO mAP@0.5:0.95 scores rise from 34.3 for YOLOv5nu and 37.3 for YOLOv8n to 38.5 for YOLOv10n, 39.5 for YOLO11n and 40.6 for YOLO12n. YOLO26n has two reported scores: 40.9 for the one-to-many head and 40.1 for the end-to-end head [1–5]. Its position relative to YOLO12n therefore depends on the head, although both scores exceed those of YOLOv8n and YOLO11n. Published CPU latencies are 80.4 ms for YOLOv8n, 56.1 ms for YOLO11n and 38.9 ms for YOLO26n. These benchmark measurements motivate a practical question: which of their advantages remain useful after fine-tuning and deployment on different data and hardware?

Floating-debris detection illustrates why this question matters. River and canal monitoring involves small, irregular objects, and inexpensive CPU hardware can constrain deployment. Applied studies often improve a YOLO baseline by adding modules. Their conclusions depend on both the baseline selected for the task and the size of the improvement relative to training and evaluation variability. A benchmark score helps select candidates; target-data comparisons establish whether a candidate offers an accuracy advantage that is large enough to affect deployment.

Doubts about both premises have a history in machine learning. A reality check of deep metric learning found that methods from 2006 and 2019 performed almost alike once comparisons were made fair [7]. In a reproduction of neural recommenders, simple, tuned baselines often outperformed six of the seven methods that could be reproduced [8]. On the statistical side, variance from data sampling exceeds

variance from weight initialization across several benchmarks [9]. Rankings in biomedical image-analysis challenges change when a single test case is removed [10, 11], and small evaluation sets produce error bars far larger than the differences routinely reported on them [12].

For YOLO generations specifically, the Roboflow100-VL benchmark found that YOLO11 did not consistently outperform YOLOv8 across 100 datasets despite its higher COCO score. Its authors suggested that newer YOLO models "might be overfitting to COCO" [13]. A benchmark of versions 5 to 11 on 33 datasets found post-YOLOv5 versions to fluctuate across domains, "sometimes failing to surpass their predecessors" [14]. In a study of YOLOv8, YOLO11 and YOLO26 on macroscopic wood images, YOLOv8 reached the highest mean accuracy [15]. A comparison of YOLO26 and YOLOv8 on PASCAL Visual Object Classes (PASCAL VOC) and VisDrone found the newer model's advantage on VOC to narrow on the small-object aerial data, where the nano model trailed YOLOv8n [16]. On 600 orchard images, a comparison of YOLOv8, YOLO11 and YOLO26 found YOLO11s, not YOLO26s, the most accurate model [17]. On the speed side, the authors of RF-DETR traced disagreements between published latency figures to graphics processing unit (GPU) power throttling [18].

Existing studies establish that a newer YOLO need not win on every dataset. Their evidence includes multi-domain comparisons [13, 14], repeated image-level splits and paired tests for wood classification [15], and recent comparisons involving YOLO26 [16, 17]. The remaining task is to connect a downstream accuracy difference to the uncertainty relevant to model selection. Training seeds can change the fitted detector, while test-image composition can change the measured gap even for fixed detectors. We examine both sources for recent lightweight detection models and distinguish stable pairwise differences from unstable complete rankings. We also evaluate the relative CPU gains between recent releases using interleaved measurements. This extends earlier x86 and ARM benchmarking through YOLOv8 [19] and complements recent edge-SoC deployment analysis [20].

Here we test whether the generational gains reported on COCO and in the published CPU tables carry over to fine-tuned lightweight detectors. We trained six nano-scale YOLO generations with three seeds each on a public floating-debris dataset [21] and a PASCAL VOC subsample [22], and three generations on a VisDrone subsample [23], each with 2,400 training, 300 validation and 300 test images. Enumerating the seed combinations measures how the ranking depends on training, and paired bootstrap resampling of the test images measures the uncertainty of each score difference. We find that the downstream order departs from the COCO order. YOLO26n leads YOLOv8n on VOC but trails it on floating debris and VisDrone, and these pairwise differences hold across seeds and resamples while the complete six-model ranking does not. The uncertainty of a comparison also depends on the dataset, not only on the number of test images.

We then timed the generations on three CPUs with randomized interleaved trials. The newer models are faster, but by a fraction of the reduction implied by the published tables. Finally, an audit of 60 floating-debris papers shows how this evidence bears on applied work: most papers build on an Ultralytics baseline, few report repeated training, and many module gains are below half a point. Together, the

results show that an upgrade or a baseline choice needs pairwise intervals on the target data and timing on the deployment processor.

Results

Accuracy advantages change direction across downstream datasets

The sizes of the generational steps also change. The published COCO gain from YOLOv8n to YOLO11n is 2.2 points, compared with − 0.2 on floating debris, + 1.8 on VOC and + 0.2 on VisDrone. From YOLO11n to YOLO26n, the two COCO head scores imply gains of 0.6–1.4 points; the downstream changes are − 1.4, + 4.1 and − 0.9 points. Thus, using either YOLO26n head as the COCO reference preserves the central contrast: the v8n–11n–26n progression improves accuracy on VOC but reverses its overall direction on floating debris. On VisDrone, the main difference is YOLO26n's deficit relative to the earlier two models.

Table 1
Published COCO scores and downstream test mAP@0.5:0.95 (%, mean ± s.d. across three training seeds). YOLO26n has a one-to-many COCO score of 40.9 and an end-to-end score of 40.1. The six-model Kendall correlations are shown for both reference scores in that order. Floating debris and VOC used the fixed training recipe; VisDrone used default optimizer selection.

Model	COCO (official)	Floating debris	VOC	VisDrone
YOLOv5nu	34.3	63.46 ± 0.15	47.35 ± 0.75	–
YOLOv8n	37.3	64.18 ± 0.12	47.03 ± 0.33	12.89 ± 0.11
YOLOv10n	38.5	63.20 ± 0.10	49.65 ± 1.29	–
YOLO11n	39.5	63.95 ± 0.03	48.87 ± 0.18	13.09 ± 0.23
YOLO12n	40.6	63.83 ± 0.05	49.79 ± 2.05	–
YOLO26n	40.9 / 40.1	62.56 ± 0.23	53.01 ± 0.79	12.20 ± 0.16
Kendall τ, six models (one-to-many / end-to-end reference)	–	−0.33 / −0.20	0.73 / 0.60	–
Kendall τ with COCO order (v8n, 11n, 26n)	–	−1.00	1.00	−0.33

Stable pairwise differences coexist with variable complete rankings

We first examined the rankings across the training runs. Taking one of the three runs for each model gives 729 combinations for six models and 27 for three. On floating debris, all 729 combinations give the

same complete order, with YOLOv8n first and YOLO26n last. YOLO26n is also last in all 27 VisDrone combinations and first in all 729 VOC combinations. Thus, the largest pairwise contrasts persist across the three seeds even where intermediate models change position.

We then applied 1,000 paired bootstrap resamples to each test set (Fig. 2). On floating debris, the three-seed mean gap of YOLOv8n minus YOLO26n is 1.62 points, with a 95% percentile interval of 1.09–2.11 points; the gap is positive in every resample. On VOC, YOLO26n minus YOLOv8n is 5.98 points, with an interval of 3.77–8.46 points. On VisDrone, YOLOv8n exceeds YOLO26n in 99.1% of the bootstrap resamples, and the interval of the gap is 0.12–1.18 points. Intermediate pairs are less distinct: YOLO11n and YOLO12n exchange order in about one quarter of floating-debris resamples, while YOLOv10n and YOLO12n remain close on VOC. The complete six-model order is retained in 55.1% and 16.4% of resamples, respectively. On floating debris, YOLOv8n remains first in 84.4%, whereas YOLO26n remains last in 99.8%. On VOC, YOLO26n remains first in 99.4%.

On the VisDrone validation split, which also selected the checkpoints, YOLO26n trails YOLOv8n by 1.2 points and YOLO11n by 1.1 points. Together, seed enumeration and test-image resampling show that the YOLO26n deficit on the two datasets persists under both sources of variability, while the full order among the other models is less stable.

The floating-debris deficit spans object sizes and persists under default optimizer selection

Two explanations for the reversal can be examined with the data in hand. The first is object size: both reversing datasets are dominated by small objects, and YOLO26 introduces a label-assignment scheme advertised to help with them [5]. A deficit confined to small objects would therefore concentrate in the smallest stratum. Stratifying the floating-debris objects by relative area and recomputing AP with an independent implementation shows the opposite pattern (Fig. 3a). YOLO26n trails YOLOv8n by 1.9 points for objects below 1% of the image (95% interval 1.35 to 2.50), by 1.0 for objects between 1% and 5% (0.31 to 1.61) and by 3.5 for objects above 5% (2.51 to 4.99). The deficit is thus present in every stratum and largest for the largest objects. On VOC, YOLO26n leads in the small stratum as well as overall, although that stratum holds only 87 objects and its interval is wide.

The second explanation is the training recipe. Our main runs used one fixed recipe for all generations (AdamW, learning rate 0.001). For a dataset of this size, the framework's default instead selects AdamW with a learning rate of 0.002. We retrained YOLOv8n and YOLO26n on the floating-debris data under the default recipe with two seeds. On the matched seeds this narrowed the gap from 1.54 to 0.89 points (1.62 over all three fixed-recipe seeds) but did not close it. All VisDrone runs had already used the default recipe (Fig. 3b). The change of recipe moved YOLO26n by 0.3 to 1.1 points and YOLOv8n by at most 0.4. A factor of two in learning rate therefore shifts the gap between two generations by about as much as the gaps that separate adjacent generations on this dataset. At the small scale YOLO26s also has the lowest mean on the floating-debris data. It is separated from YOLOv10s and YOLOv8s, however, only with probabilities of 0.74 and 0.88, and YOLO11s, not YOLOv8s, ranks first (Supplementary Fig. S1).

The official CPU speedups reproduce in direction but not in size

We timed the generations on three CPUs with randomized interleaved trials (Table 2). Platform A is a desktop AMD Ryzen 5 5600G, platform B a two-core cloud Intel Xeon at 2.8 GHz and platform C a four-core cloud Intel Xeon at 2.2 GHz, the last of the processor class named in the official claim. For the YOLOv8n–YOLO11n–YOLO26n steps, the newer model is faster or level on every platform, but the step falls short of the official figure (Fig. 4a). From YOLOv8n to YOLO11n the official tables imply a 30% reduction in latency. We measured 1.1% on platform A (95% interval over the ten paired rounds − 1.0 to 3.3%), 7.3% on platform B and 8.2–12.1% on platform C (6.1 to 14.9%). From YOLO11n to YOLO26n the official tables imply 31%, which is the "43% faster" of the documentation expressed as a reduction in latency. On platforms A, B and C we measured 14.1% (12.1 to 16.0%), 22.1% and 16.7–20.2% (15.4 to 23.1%). On platform A the oldest generation, YOLOv5nu, cannot be told apart from YOLO26n (34.26 against 34.37 ms, paired-round interval of the difference − 3.0 to 2.1%). The official tables nonetheless put its latency at nearly twice that of YOLO26n.

Table 2
Mean ONNX CPU latency (ms) under randomized interleaved timing (1,000 inferences per model, batch size 1, 640 × 640, FP32). Official values are from the documentation [1–5], which gives no CPU latency for YOLOv10n and YOLO12n. Platform A ran Windows 10 with ONNX Runtime 1.23.2, platform B ONNX Runtime 1.25.0 and platform C ONNX Runtime 1.23.2; platforms A and B timed the floating-debris weights.

Model	Official	A: Ryzen 5 5600G, 6 cores	B: Xeon 2.8 GHz, 2 vCPU	C: Xeon 2.2 GHz, 4 vCPU, COCO weights	C: debris weights
YOLOv5nu	73.6	34.3	37.8	100.9	93.4
YOLOv8n	80.4	40.5	44.9	111.6	106.0
YOLOv10n	–	38.3	41.5	101.6	97.0
YOLO11n	56.1	40.0	41.6	98.0	97.2
YOLO12n	–	49.0	50.5	118.7	117.7
YOLO26n	38.9	34.4	32.4	81.6	77.4

On platform C, the speedups obtained with official COCO weights and fine-tuned weights differ by at most four percentage points. Fixed-order timing changes the mean latencies by less than 5%, but includes isolated slow rounds of up to 150 ms (Supplementary Fig. S2). A separate eight-image validator test illustrates how short measurements can vary: across three repetitions, the slowest inference estimate for a model is 1.13–1.60 times its fastest, and the six-model order changes. Randomized interleaved measurement gives round-level standard errors of 0.5–1.7 ms (Fig. 4b). The randomized rankings on platforms A and B agree more closely than their fixed-order rankings (Spearman correlations 0.943 and 0.771).

Static efficiency proxies and end-to-end costs provide two further checks. Parameter count, FLOPs, T4 GPU latency and ONNX node count have Spearman correlations of 0.09–0.71 with CPU latency across the three platforms (Supplementary Fig. S3). These descriptors do not consistently reproduce the measured CPU order across the six models. On platform A, decoding, resizing and post-processing add 17.2–17.6 ms per frame; inference accounts for 64–72% of end-to-end latency. The 38.8% inference gap between the slowest and fastest nano model narrows to 24.1% end to end (Supplementary Fig. S4).

Uncertainty depends on the dataset and evaluation size

Table 3 separates the pooled seed standard deviation from the median standard deviation of paired test-image differences for the two six-model datasets. Combining twice the pooled seed variance with the paired test-sampling variance gives a descriptive scale for comparing two single runs. It is larger on VOC than on floating debris at every evaluation size. This separates two practical needs: repeated training describes variability in the fitted models, while paired resampling describes variability in their score differences on the available evaluation images.

Table 3
Components of variability in mAP@0.5:0.95 (percentage points). Seed s.d. is pooled across six models, each with three seeds. Paired test s.d. is the median across 15 model pairs at each resampling size. Combined s.d. is the square root of twice the seed variance plus the paired test variance. Values describe these experiments.

Test images	Debris seed s.d.	Debris paired test s.d.	Debris combined s.d.	VOC seed s.d.	VOC paired test s.d.	VOC combined s.d.
50	0.13	0.71	0.74	1.09	4.27	4.54
100	0.13	0.50	0.53	1.09	3.21	3.56
150	0.13	0.40	0.44	1.09	2.66	3.08
200	0.13	0.36	0.40	1.09	2.25	2.73
300	0.13	0.29	0.34	1.09	1.73	2.32

At 300 resampled images, the median paired-difference standard deviation is 0.29 points on floating debris and 1.73 on VOC. The corresponding combined scales are 0.34 and 2.32 points. With 50 resampled images they rise to 0.74 and 4.54 points. Equal image counts therefore give quite different precision for model comparison. A leave-one-seed-out check also compares the bootstrap frequency from one seed with the ordering on the other two seeds. Across 135 pair-by-seed prediction cases, the mean squared error against the fraction of the two held-out seeds preserving the order is 0.067, compared with 0.100 for always predicting retention. Against the individual binary outcomes, the Brier score is 0.100, compared with 0.133 for always predicting retention and 0.250 for a probability of 0.5 (Supplementary Fig. S5).

Applied papers report small module gains with limited repetition information

The documentary audit contains 60 floating-debris detection papers (Fig. 5). Of the 57 papers with an identifiable baseline, 49 use an Ultralytics model. Of the 42 papers read in full, two explicitly report repeated training for the main comparison, with three and five runs. The other 40 do not state how many runs were made. Thirty-six papers have readable one-module ablation contrasts. Their smallest non-negative gains are below 0.5 points in 23 cases, and ten tables also contain at least one negative one-module step.

Discussion

The results refine the meaning of a generational improvement in lightweight object detection. A higher COCO score can lead to a downstream gain, as it does for YOLO26n on VOC, or to a deficit, as it does on floating debris and VisDrone. In these experiments the contrast is larger than the within-configuration seed spread and retains its sign under paired test-image resampling. At the same time, the complete order among intermediate models is much less stable. Accuracy comparison is therefore most informative when it identifies the model pairs that can be distinguished on the target task.

Earlier multi-dataset comparisons documented variation in the winning generation [13–17]. The contribution here is to connect that variation to two distinct uncertainty sources for lightweight detection, and to examine the associated CPU gains. The floating-debris result illustrates the distinction: all 729 seed combinations preserve the six-model order, but only 55.1% of image resamples preserve it. YOLOv8n remains first in 84.4%, while the YOLO26n deficit is much more stable. Reporting only the winner or only a mean ranking would obscure this difference. Pairwise intervals and resampling frequencies make the evidence relevant to a concrete choice between models.

The size and recipe analyses sharpen the downstream finding. YOLO26n's floating-debris deficit occurs in every relative-area stratum and is largest for the largest objects. Small-object composition alone does not account for the pattern in this dataset. Default optimizer selection narrows the matched-seed gap from 1.54 to 0.89 points, showing that training settings materially affect its size. The two contrasting application domains—water-surface debris and aerial road users—also differ from VOC in image content and category composition. This motivates a broader study of which task characteristics interact with model generation. The present results locate a reproducible comparison problem for that work rather than selecting a specific architectural cause.

CPU comparison adds a separate deployment lesson. The reported latency tables imply larger relative gains than those we measured on three platforms. Interleaving puts models in a shared measurement session and distributes changes in system conditions across their trials [24–26]. On platform C, the relative gains remain similar for COCO and fine-tuned weights, while the eight-image validator test varies appreciably across repeats. These observations complement GPU measurements affected by power throttling [18], CPU benchmarks of earlier YOLO generations [19] and recent edge-SoC studies [20]. They support measuring upgrade gains on the deployment platform with a protocol long enough to characterize variability.

For engineering teams, an upgrade decision should combine the target-task accuracy gap, its uncertainty and the measured end-to-end cost. A paired interval that excludes zero provides evidence for an accuracy difference; its magnitude and the application's tolerance determine whether the difference matters. Repeated seeds show whether this evidence depends on a particular fitted model. For applied research, the same comparison helps choose a suitable baseline and evaluate a module increment. The audit shows why reporting the number of runs and their spread is useful even when a module gain is numerically small. Framework release tables would become more useful for this purpose by recording the evaluation head and by reporting accuracy variability and contemporaneous timing across generations. Supplementary Table S2 summarizes these practices.

The study evaluates transfer under a common small-data budget: 2,400 training images per dataset, three seeds per main configuration and a 100-epoch maximum. Equal image counts leave category diversity and object density different, as the uncertainty scales demonstrate. Per-generation tuning, longer training and alternative data splits may change the gaps. The bootstrap intervals quantify image-sampling uncertainty for the observed models; the combined variability scale assumes independent seed errors and uses pooled variances, so it serves as a descriptive reference rather than a calibrated probability of future ordering. CPU measurements concern batch-one FP32 ONNX inference, with runtime versions and available virtual cores differing across platforms. The short-validator experiment characterizes that local protocol, not the provenance of the published latency tables. Image-level splits also leave possible near-duplicate or sequence dependence to be checked against the source images.

Generational benchmark scores are useful evidence when paired with target-task uncertainty and deployment measurement. The practical outcome of this study is a comparison workflow that separates stable accuracy contrasts from variable rankings and measures the speed gain available on the chosen CPU. It turns a release-level improvement into a testable model-choice question.

Methods

Official figures

COCO mAP@0.5:0.95 and CPU ONNX latency were taken from the Ultralytics documentation accessed in September 2026 [1–5]. CPU figures are available for YOLOv5nu (73.6 ms), YOLOv8n (80.4 ms), YOLO11n (56.1 ms) and YOLO26n (38.9 ms). We calculate relative reductions between published means. YOLO26n has a one-to-many COCO score of 40.9 and an end-to-end score of 40.1; both are used as reference values. Our validation scripts set no head override. In Ultralytics 8.4.106 the YOLO26 model configuration enables the end-to-end head, and the validator evaluates that head unless an override is given, so the downstream YOLO26n scores correspond to the 40.1 reference. We nonetheless compare the COCO rank association under both published head scores. The six-model correlations change, while comparisons among YOLOv8n, YOLO11n and YOLO26n retain the same reference order. The timed YOLO26n export uses the NMS-free output with 300 detections per image.

Datasets

The floating-debris dataset is IWHR_AI_Label_Floater_V1, with 3,000 images and one class [21]. An 8:1:1 split with seed 42 gives 2,400, 300 and 300 images, containing 18,903, 2,442 and 2,344 objects. Across the complete split, 53.9% of objects occupy less than 1% of image area. The VOC subsample uses the framework's PASCAL VOC conversion [22]: 2,400 training and 300 validation images from the 2007 + 2012 trainval pool, and 300 images from VOC2007 test, with 716 test objects in 20 classes. The VisDrone subsample uses the converted VisDrone2019-DET data [23]: 2,400 images from train, 300 from validation and 300 from test-dev. After excluding the others category and ignored regions, the splits contain 124,785, 21,395 and 13,377 objects in ten classes. Across all three VisDrone splits, 97.5% occupy less than 1% of image area. Each dataset has the same image budget and split ratio; category diversity and object counts remain dataset-specific. The released scripts regenerate VOC and VisDrone subsamples with seed 42, and the floating-debris file manifest is provided. Split fingerprints are EF56F1F0F7B28F68, EACC0A312D60CBF6 and 181CB684E5D45AF2, respectively. They identify the image-name manifests.

Training

All models were trained with Ultralytics 8.4.106 from COCO-pretrained weights on NVIDIA Tesla T4 GPUs. Each run used 640 × 640 input, batch size 16, default augmentation, deterministic mode and one of the seeds 42, 1 and 2 [27]. Training lasted at most 100 epochs and stopped early after 20 epochs without improvement. On the floating-debris and VOC data, all six nano generations were trained with a fixed recipe, AdamW with an initial learning rate of 0.001, so that every generation saw the same optimizer settings (36 runs). To check that the floating-debris result did not depend on this choice, we retrained YOLOv8n and YOLO26n there with the framework's default optimizer selection. For this dataset size, the default selects AdamW with a learning rate of 0.002 for both models (4 runs, seeds 42 and 1), so the MuSGD optimizer introduced with YOLO26 [5] was not used. On VisDrone, YOLOv8n, YOLO11n and YOLO26n were trained with the default selection throughout, which chose AdamW with a learning rate of 0.000714 (9 runs). The optimizer and learning rate actually used were read back from each trainer rather than assumed. For each run we kept the checkpoint with the best validation fitness, and the test set was used only for reporting. Small-scale models of the six generations were also trained on the floating-debris data with the same fixed recipe and three seeds (18 runs, Supplementary Fig. S1). This gives 67 training runs in all, of which the 49 nano-scale runs on the three datasets form the main comparison.

Accuracy evaluation and noise

Test mAP@0.5:0.95 was computed with the framework's evaluator (confidence threshold 0.001, NMS IoU 0.7). During evaluation we intercepted the per-image records that the evaluator accumulates: the true-positive matrix over ten IoU thresholds, the confidence and class of each detection, and the target classes. Recomputing mAP from these records reproduced the reported value to within 10^{-6} for every

run on the full test set. To estimate test-set sampling noise, we resampled test images with replacement 1,000 times and recomputed mAP from the intercepted records of the drawn images. Every model used the same draws, so differences between models are paired. Seed noise was captured by enumerating all combinations of one run per model and recording the order each combination produces. A reported bootstrap frequency that one model exceeds another is the fraction of resamples in which the seed-mean score of the first exceeds that of the second. Intervals are the 2.5th and 97.5th percentiles, and agreement between the official order and a downstream order is measured by Kendall's τ. These intervals capture test-image variation for the trained runs, and seed-combination enumeration covers the three training seeds. The all-pair displays are descriptive, and their intervals are not adjusted for multiple comparisons.

Size-stratified accuracy

To test whether YOLO26n's deficit is concentrated on small objects, we recorded the ground-truth and predicted boxes of every test image during evaluation. We then recomputed AP with an independent implementation, pycocotools. Box coordinates were normalized to the original image and rescaled to a virtual 1,000 × 1,000 frame, so that COCO's area ranges become relative-area ranges. Strata were below 1%, 1–5% and above 5% of the image area, and the 100 highest-confidence detections per image were retained. Across all 36 floating-debris and VOC runs, overall AP from pycocotools correlated with the framework's mAP at r = 0.994 and 1.000 respectively. Stratum differences between YOLOv8n and YOLO26n were bootstrapped with 100 paired resamples of test images.

Latency

Models were exported to static FP32 ONNX graphs (1 × 3 × 640 × 640) with the same framework version. They were run with ONNX Runtime's CPU execution provider at default thread settings. Platform A was a desktop AMD Ryzen 5 5600G (6 cores, 12 threads) with Windows 10 and ONNX Runtime 1.23.2. Platform B was a cloud instance with two virtual cores of an Intel Xeon at 2.8 GHz, running Linux and ONNX Runtime 1.25.0. Platform C was a cloud instance with four virtual cores of an Intel Xeon at 2.2 GHz, running Linux and ONNX Runtime 1.23.2. Platforms A and B timed the floating-debris models, and platform C timed both the official COCO weights and the floating-debris models. We used randomized multiple interleaved trials [24]. Each model first ran 30 warm-up inferences, followed by ten rounds in which every model performed 100 consecutive inferences, in an order reshuffled before the round. This gave 1,000 timings per model, which we summarize by the mean and by the standard error of the ten round means. The same models were also timed in a fixed order for comparison. As a short-measurement comparison, platform C also ran the framework's validator for each ONNX model on its eight-image COCO sample with batch size one. These runs were repeated three times in random order, and we recorded the reported inference time. On platform A, end-to-end latency was decomposed into decoding, pre-processing, inference and post-processing (confidence 0.25, NMS IoU 0.7 where required). This decomposition used 100 floating-debris images of mean resolution 1976 × 1140 under the same

interleaved protocol. Spearman correlations between CPU latency and four static proxies (parameters, FLOPs, T4 GPU latency, ONNX node count) were computed per platform.

Variability scale and leave-one-seed-out check

For Table 3, the seed standard deviation is pooled across the six nano models within each dataset. For each evaluation size, we use the median standard deviation of the paired score difference across 500 image resamples over the 15 model pairs. The descriptive combined standard deviation is the square root of twice the pooled seed variance plus the median paired-difference variance. The paired-difference component is based on seed-mean scores; combining it with single-run seed variance provides an approximate reference scale. All components are reported separately. For the held-out check, each seed supplies the bootstrap frequency that preserves a model pair's observed order. The other two seeds supply the fraction retaining that order and two individual binary outcomes. We compute mean squared error for the fractional outcome and the Brier score for the binary outcomes, with always-retain and probability-0.5 reference predictions. The 135 pair-by-seed cases come from floating-debris nano, floating-debris small and VOC nano groups. The cases share models and seeds, so the check is internal to this study.

Literature audit

The audit used a Crossref search conducted on 30 September 2026 for journal articles from 1 January 2021 to 30 September 2026. Ten queries combined YOLO with terms for floating debris, garbage, waste, litter, trash and floating objects on water surfaces or rivers. Keeping up to 600 relevant records per query gave 2,444 unique records. Title screening retained 130 records, from which 53 eligible papers were identified; seven from an earlier scoping search gave 60. Eligibility required water-surface debris detection and a modified YOLO detector compared with a YOLO baseline; conference proceedings, underwater or satellite detection, unmodified-model comparisons and non-YOLO architectures were excluded. For each record we coded access status, baseline identity, repetition evidence, evaluation size, ablation metric and one-module contrasts. A run count is coded only when the source states it explicitly. A single reported score, a default seed or the absence of a repetition statement is coded as unknown. Forty-two records have full-text extraction, 16 have abstract-only information and two lack readable text. Abstract-only records contribute to baseline descriptions. For 36 readable ablation tables, the smallest non-negative difference is taken between rows differing by one module, and negative steps are recorded separately. Cumulative tables use the increment over the previous row. Where available, module-alone increments relative to the baseline are also recorded. Test counts derived from totals and split ratios remain marked as derived, and validation-image counts are identified as such. For 23 of these records, a second language-model extraction supplies a consistency check; quoted rows and disagreements are included in Supplementary Data 1.

All figures were drawn with Matplotlib from the stored result files, and the scripts that produce them from the raw records are included in Supplementary Data 1.

Use of large language models

Claude (Anthropic) assisted with training, evaluation, timing and plotting scripts, literature screening and extraction, and drafting the English manuscript. ChatGPT (OpenAI) assisted with revising the manuscript text, consistency checks, and the plotting and reproduction scripts. Literature extraction was repeated as a language-model-assisted consistency check, not as independent human coding. The research questions and analyses were selected by the author, who is responsible for checking the manuscript, code and extracted source evidence and for the final content.

Declarations

Competing interests:

The author declares no competing interests.

Funding

This research did not receive funding.

Author Contribution

G.Y. conceived the study, designed and ran the experiments, analysed the data, and prepared and revised the manuscript with the assistance described in Methods.

Data Availability

The floating-debris dataset is available from its original publication (Qiao, Yang & Wang, Sci. Data 12, 385, 2025; https://doi.org/10.1038/s41597-025-04594-9), PASCAL VOC from Everingham et al. (https://doi.org/10.1007/s11263-009-0275-4) and VisDrone from Zhu et al. (https://doi.org/10.1109/TPAMI.2021.3119563). Supplementary Data 1, a ZIP archive uploaded with this article, contains the file list of our floating-debris split, per-run metrics, stored bootstrap score arrays, detection records, CPU timing data, literature-audit records, and all training and analysis scripts, including a script that recomputes the main reported numbers. The six floating-debris nano ONNX models for seed 42 are available from the author on request.

Code availability

The scripts for training, bootstrap resampling, seed-combination enumeration, size-stratified evaluation, randomized interleaved timing, the literature search and the figures are included in Supplementary Data 1, together with a script that recomputes the main reported numbers from the released records.

References

1. Ultralytics. YOLOv5 (Ultralytics documentation). September (2026). https://docs.ultralytics.com/models/yolov5/ (accessed.
2. Ultralytics. YOLOv8 (Ultralytics documentation). September (2026). https://docs.ultralytics.com/models/yolov8/ (accessed.
3. Ultralytics & YOLOv10. September. and YOLO12 (Ultralytics documentation). (2026). https://docs.ultralytics.com/models/yolov10/; https://docs.ultralytics.com/models/yolo12/ (accessed.
4. Ultralytics. YOLO11 (Ultralytics documentation). September (2026). https://docs.ultralytics.com/models/yolo11/ (accessed.
5. Ultralytics. YOLO26 (Ultralytics documentation). September (2026). https://docs.ultralytics.com/models/yolo26/ (accessed.
6. Lin, T. Y. et al. Microsoft COCO: common objects in context. In Computer Vision – ECCV 2014, Lecture Notes in Computer Science 8693, 740–755 (2014). https://doi.org/10.1007/978-3-319-10602-1_48
7. Musgrave, K., Belongie, S. & Lim, S. N. A metric learning reality check. In Computer Vision – ECCV 2020, 681–699 (Springer, 2020). https://doi.org/10.1007/978-3-030-58595-2_41
8. Ferrari Dacrema, M., Cremonesi, P. & Jannach, D. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In Proc. 13th ACM Conference on Recommender Systems, 101–109 (2019). https://doi.org/10.1145/3298689.3347058
9. Bouthillier, X. et al. Accounting for variance in machine learning benchmarks. In Proc. Machine Learning and Systems 3 arXiv:2103.03098 (2021).
10. Maier-Hein, L. et al. Why rankings of biomedical image analysis competitions should be interpreted with care. *Nat. Commun.* **9**, 5217. https://doi.org/10.1038/s41467-018-07619-7 (2018).
11. Wiesenfarth, M. et al. Methods and open-source toolkit for analyzing and visualizing challenge results. *Sci. Rep.* **11**, 2369. https://doi.org/10.1038/s41598-021-82017-6 (2021).
12. Varoquaux, G. Cross-validation failure: small sample sizes lead to large error bars. *NeuroImage* **180**, 68–77. https://doi.org/10.1016/j.neuroimage.2017.06.061 (2018).
13. Popov, M. et al. Roboflow100-VL: a multi-domain object detection benchmark for vision-language models. *Adv. Neural. Inf. Process. Syst.* **38**, 12035–12073. https://doi.org/10.52202/085713-0363 (2025).
14. Jiang, T. & Zhong, Y. ODverse33: is the new YOLO version always better? A multi domain benchmark from YOLO v5 to v11. (2025). arXiv:2502.14314.
15. Spancerski, J. S., de Paula Filho, P. L., Kugler, M. & Schneider, F. K. A comprehensive benchmark of multi-generation YOLO architectures for forest species identification from macroscopic wood images. *Sci. Rep.* **16**, 25802. https://doi.org/10.1038/s41598-026-54925-y (2026).
16. Oguine, C. G., Oguine, K. J., Oguine, O. M. & Oguine, O. C. YOLO26 vs. YOLOv8: a comprehensive architectural benchmark of next-generation real-time object detection models. (2026).

arXiv:2605.24831.

17. Sapkota, R., Karkee, M. & Optimizing YOLO27, YOLO26, YOLO11, and YOLOv8 for fine-grained small-object detection and segmentation in complex orchard environments. arXiv:2608.23636 (2026).
18. Robinson, I., Robicheaux, P., Popov, M., Ramanan, D. & Peri, N. RF-DETR: neural architecture search for real-time detection transformers. In Proc. International Conference on Learning Representations arXiv:2511.09554 (2026).
19. Lazarevich, I. et al. YOLOBench: benchmarking efficient object detectors on embedded systems. In Proc. IEEE/CVF International Conference on Computer Vision Workshops, 1161–1170 (2023). https://doi.org/10.1109/ICCVW60793.2023.00126
20. Kong, C. et al. Object detection on low-compute edge SoCs: a reproducible benchmark and deployment guidelines. *Sci. Rep.* **16**, 5875. https://doi.org/10.1038/s41598-026-36862-y (2026).
21. Qiao, G., Yang, M. & Wang, H. An annotated dataset and benchmark for detecting floating debris in inland waters. *Sci. Data*. **12**, 385. https://doi.org/10.1038/s41597-025-04594-9 (2025).
22. Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. The PASCAL visual object classes (VOC) challenge. *Int. J. Comput. Vis.* **88**, 303–338. https://doi.org/10.1007/s11263-009-0275-4 (2010).
23. Zhu, P. et al. Detection and tracking meet drones challenge. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7380–7399. https://doi.org/10.1109/TPAMI.2021.3119563 (2022).
24. Abedi, A. & Brecht, T. Conducting repeatable experiments in highly variable cloud computing environments. In Proc. 8th ACM/SPEC International Conference on Performance Engineering, 287–292 (2017). https://doi.org/10.1145/3030207.3030229
25. Georges, A., Buytaert, D. & Eeckhout, L. Statistically rigorous Java performance evaluation. In Proc. 22nd ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications, 57–76 (2007). https://doi.org/10.1145/1297027.1297033
26. Mytkowicz, T., Diwan, A., Hauswirth, M. & Sweeney, P. F. Producing wrong data without doing anything obviously wrong! In Proc. 14th International Conference on Architectural Support for Programming Languages and Operating Systems, 265–276 (2009). https://doi.org/10.1145/1508244.1508275
27. Jocher, G., Qiu, J., Chaurasia, A. & Ultralytics, Y. O. L. O. version 8.4.106 (software). https://github.com/ultralytics/ultralytics

Figures

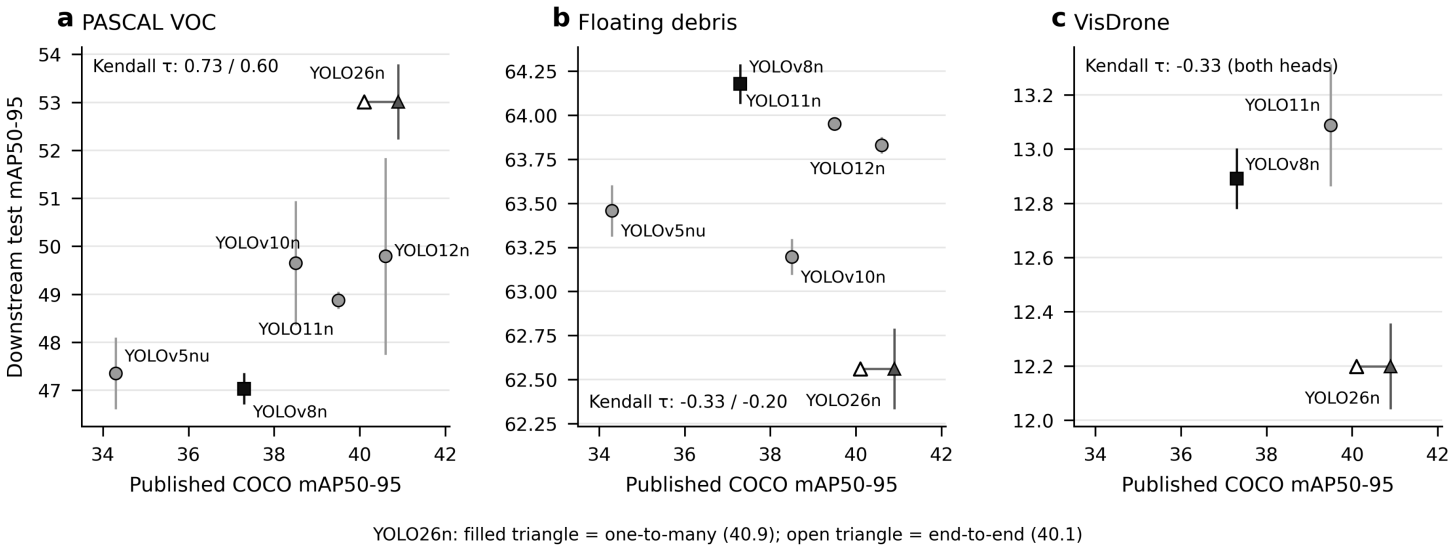


Figure 1

Published COCO scores and downstream accuracy. Points show three-seed means with seed standard deviations. YOLO26n is shown at both published head scores, 40.9 (one-to-many) and 40.1 (end-to-end); the connecting horizontal segment represents the change of reference head. Rank correlations are reported for both reference values where they differ. YOLOv8n and YOLO26n have distinct marker shapes. Panels show VOC, floating debris and VisDrone; the last includes three generations.

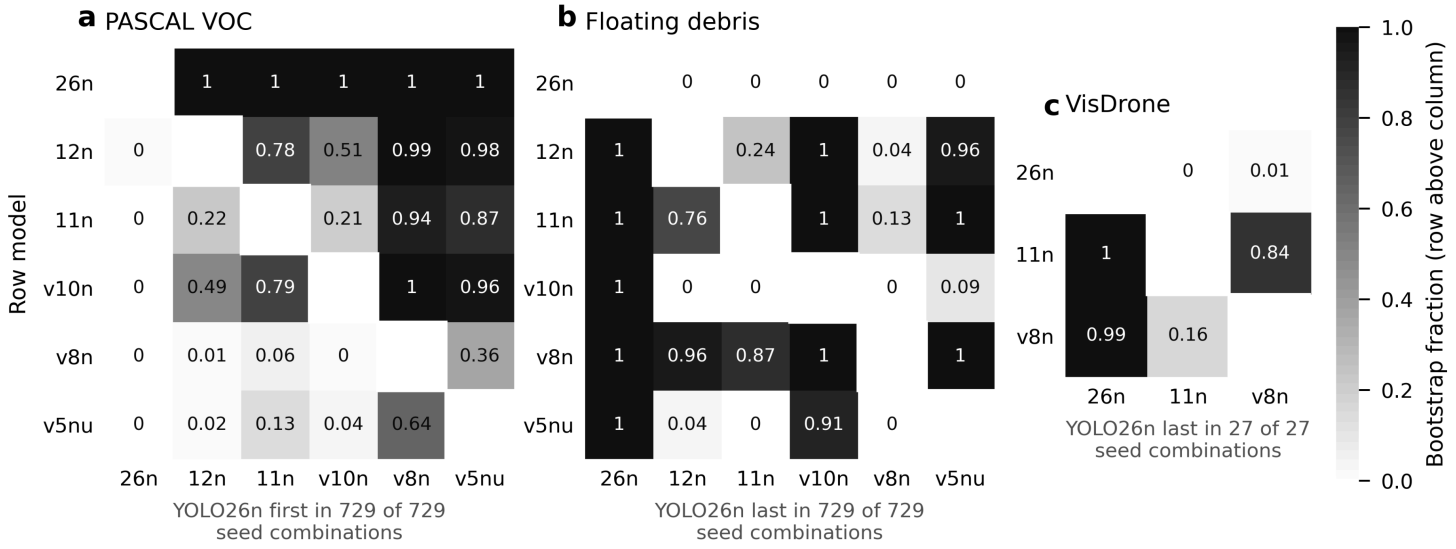


Figure 2

Pairwise accuracy stability under test-image resampling. Each cell is the fraction of 1,000 paired image bootstrap resamples in which the row model's three-seed mean exceeds the column model's mean. These frequencies hold the trained models fixed. Models are arranged by release generation, with the newest first. The notes below the panels enumerate combinations of the three training seeds.

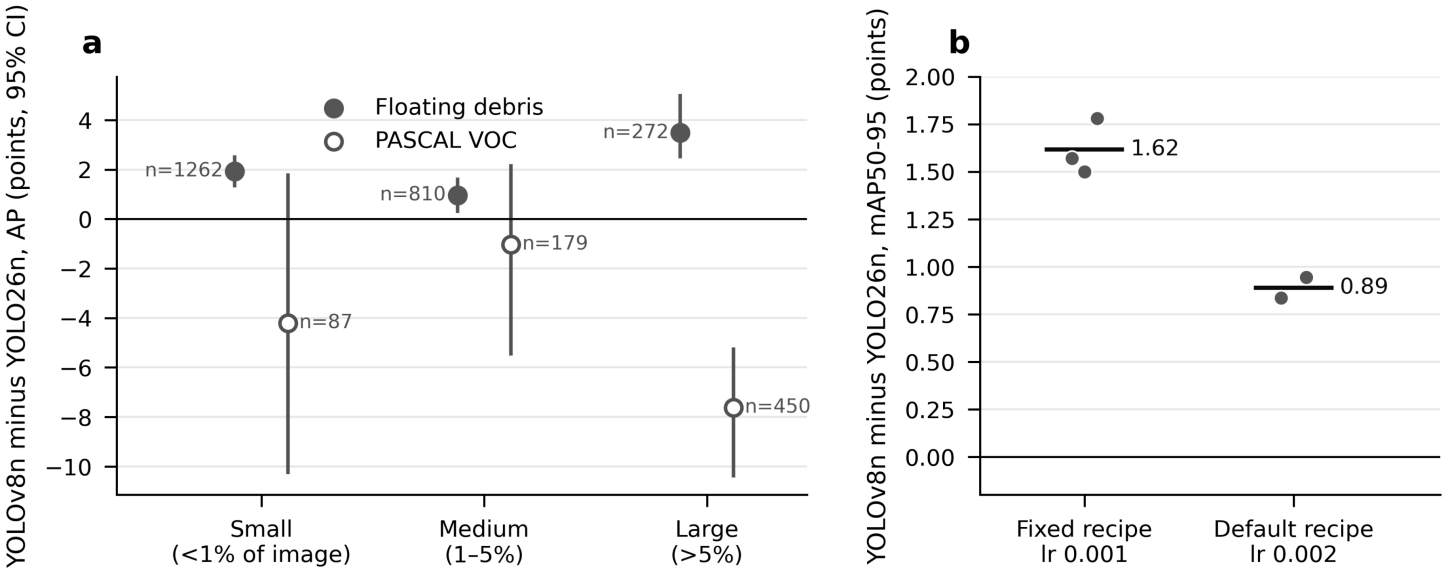


Figure 3

YOLO26n's deficit on the floating-debris data is present in every object-size stratum, largest for large objects, and persists under the framework's default training recipe.(a) Difference in AP between YOLOv8n and YOLO26n by relative object area, recomputed with pycocotools (seed means with 95% bootstrap intervals, n = number of ground-truth objects). Positive values favour YOLOv8n. (b) Difference in test mAP@0.5:0.95 under the fixed recipe used for all main runs (three seeds) and under the framework's default optimizer selection (two seeds). Bars are means.

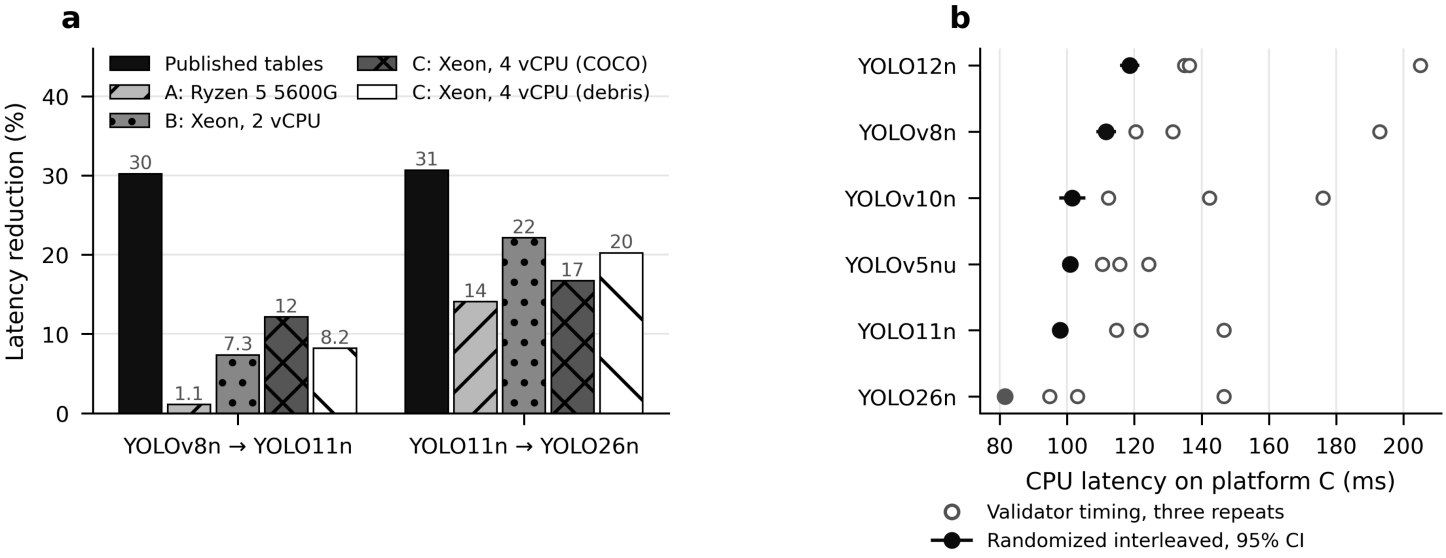


Figure 4

Official CPU speedups between YOLO generations keep their sign but lose most of their size on three CPUs. (a) Reduction in mean ONNX CPU latency from one generation to the next, as implied by the official tables and as measured with randomized interleaved timing on platforms A–C. On platform C,

both the official COCO weights and the floating-debris weights were timed. (b) On platform C, three repeats of the framework's validator timing on its eight-image sample (open circles) scatter more widely than the differences between generations. Randomized interleaved timing (filled points, 95% confidence intervals of round means) separates the generations.

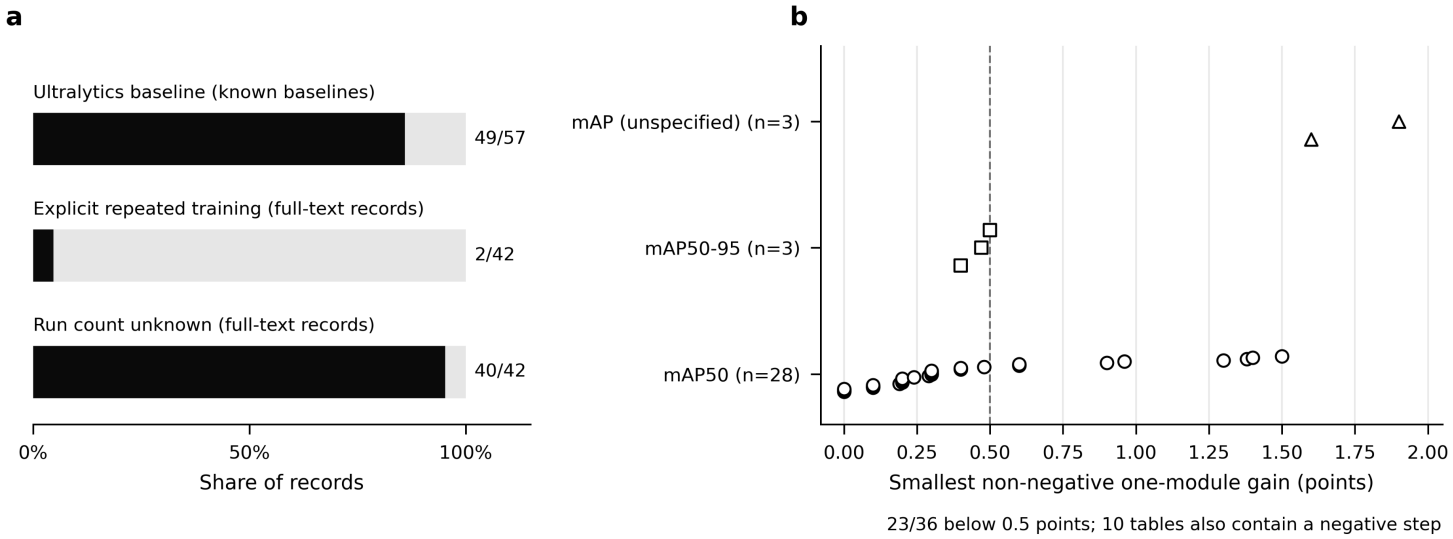


Figure 5

Baseline, repetition and module-gain reporting in the floating-debris literature. (a) Baseline identity is available for 57 of 60 records; 49 identify an Ultralytics model. Of the 42 papers read in full, two explicitly report repeated training for the main comparison and 40 do not state the run count. (b) Distribution of the smallest non-negative one-module gain across 36 readable ablation tables, shown separately by reported metric. Gains are in percentage points. Twenty-three are below 0.5 points; ten tables additionally contain a negative one-module step.

Supplementary Files

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

- SupplementaryInformationYang.pdf
- SupplementaryData1Yang.zip