Supplementary experiments and implementation details

ProtoRefine: Prototype-conditioned refinement of strawberry instance masks with native-area decoding

We provide additional evidence on local mask reconstruction, the interaction between representation and decoding, and the conditions under which an inherited foreground count helps or hinders contour correction. Complete strawberry test and development results accompany these analyses, followed by the data, training and evaluation details needed to reproduce them.

1 Complete StrawDI test results

Local refinement and higher-resolution inference address different sources of error. Table S1 retains both references: refinement improves the native 640-pixel masks, while native 896-pixel inference remains stronger in overall AP and small-object AP. Free decoding is also retained because the area constraint does not improve every overlap metric. The second YOLO26s-seg model provides the same comparisons with a different fixed set of weights.

Method	AP	AP50	AP75	AP90	AP small	AP medium	AP large
Main YOLO26s-seg model							
Native (640)	82.7182	96.9833	87.8393	66.3613	47.1412	87.0325	98.6632
Free refinement	83.9114	96.3246	89.0419	70.2881	48.0221	88.9401	98.6632
ProtoRefine	84.0339	97.1809	88.4736	69.2580	49.1810	88.5867	98.6632
Residual + native-area	83.9701	97.1809	88.5271	69.1419	49.4527	88.3651	98.6632
Native (896)	84.8135	97.5485	89.9177	69.4357	51.9339	88.8888	98.9440
Second YOLO26s-seg model							
Native (640)	82.6260	97.2265	88.2437	66.2291	46.8785	86.8958	98.7169
Free refinement	84.1092	96.5234	89.1494	69.4424	48.6441	89.1466	98.7169
ProtoRefine	83.9313	97.2265	88.5835	69.2752	49.4878	88.4542	98.7169
Residual + native-area	83.9857	96.7534	89.1566	69.0745	49.6157	88.3201	98.7169
Native (896)	85.0149	97.5993	90.5198	69.5127	52.6999	88.6837	99.1996
Other complete systems							
YOLO11s-seg (640)	82.2752	96.6251	86.9495	65.4843	45.8806	86.4787	98.8415
RTMDet-Ins-tiny (640)	80.1005	95.2574	84.9819	64.1686	38.6500	85.2937	98.3462
RTMDet-Ins-tiny (896)	83.5989	96.5304	88.3300	68.5103	47.4415	88.3790	98.9849

Table S1 Complete mask metrics on the 200 official test images. Parentheses give input size; both YOLO26s-seg models share the same architecture.

2 Representation and decoding controls

The coarse logit already combines prototypes and instance coefficients. Table S2 tests whether exposing those factors explicitly improves reconstruction beyond masked-input and near-capacity-matched coarse controls.

Method	AP	AP50	AP75	AP90	AP small	AP medium	AP large
Native (640)	83.1275	97.3186	89.6280	64.3380	50.0880	86.7528	98.5467
Full-input and point-based refinement							
Full / free	83.9342	96.6397	90.1751	67.4457	49.5083	88.5078	98.5467
Full / native-area	84.5238	97.6434	90.4975	68.4022	52.4034	88.6743	98.5467
Point adaptation	83.9200	96.8875	90.1455	68.3781	49.5356	88.5144	98.5467
Coarse-input controls							
Masked coarse / free	83.2911	96.9310	88.7900	67.3189	47.1829	87.9070	98.5467
Wide coarse / free	83.3828	96.7289	89.0180	67.3781	47.2848	88.0706	98.5467
Masked coarse / native-area	84.1917	97.6118	90.5043	67.4598	51.5994	88.4081	98.5467
Wide coarse / native-area	84.3820	97.6118	90.5129	68.4222	51.8755	88.6971	98.5467

Table S2 Representation and decoding controls on the 265 development images. Full/native-area is ProtoRefine; masked and wide coarse heads use only logits and coordinates.

Explicit factors and decoding. At seed 42, native-area decoding reduces the full–coarse AP gap from 0.55 to 0.14 points. Section 4.1 repeats this comparison across initializations.

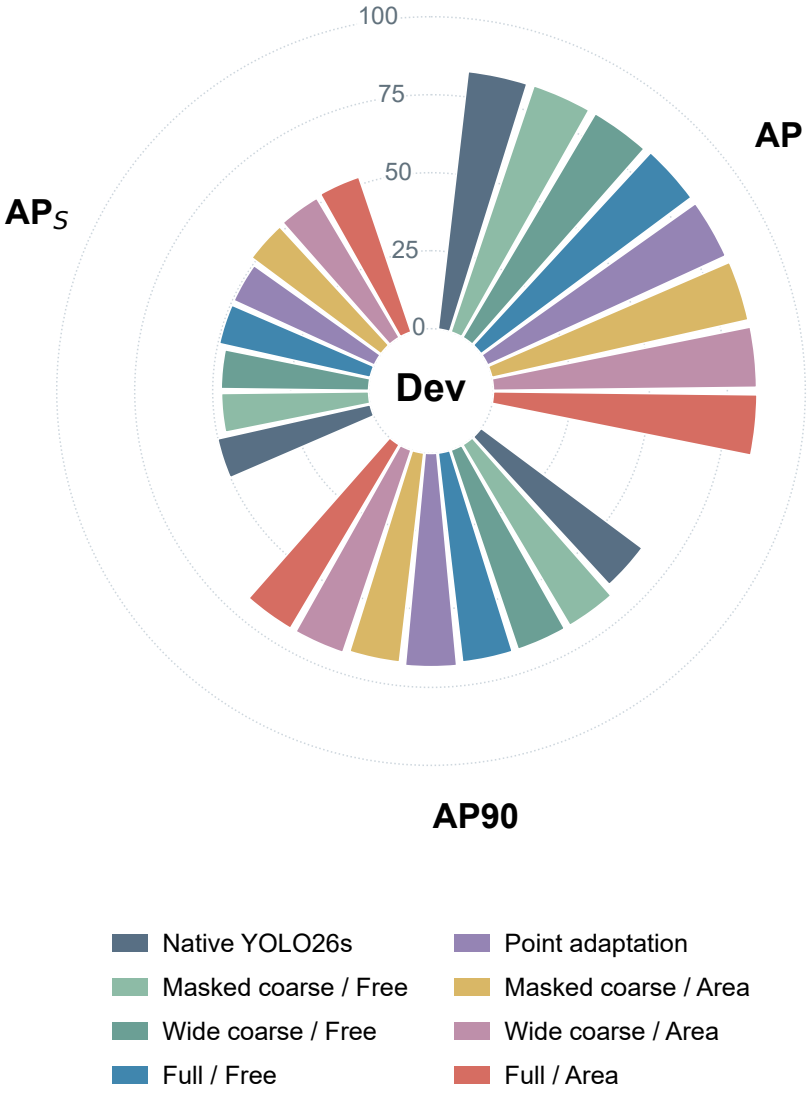


Fig. S1 Representation and decoding comparison. AP, AP90 and small-object AP for the eight controls in Table S2, on the 0–100 scale.

3 Paired test comparisons

Table S3 separates within-detector controls from complete-system references. Native-area versus free decoding remains inconclusive in AP for both fixed models.

Metric	Main YOLO26s-seg model Difference (pp) [95% interval]	Second YOLO26s-seg model Difference (pp) [95% interval]
(a) Within-detector refinement controls		
ProtoRefine (640) − YOLO26s-seg (640)		
AP	+1.3157 [+1.0206, +1.5120]	+1.3053 [+1.1289, +1.6581]
AP50	+0.1977 [-0.0862, +0.4484]	+0.0000 [+0.0000, +0.0000]
AP75	+0.6343 [+0.1055, +1.4528]	+0.3398 [+0.0378, +1.0293]
AP90	+2.8967 [+1.9023, +3.8815]	+3.0461 [+1.9224, +4.7490]
AP small	+2.0398 [+1.2940, +2.8101]	+2.6093 [+2.0202, +3.3077]
AP medium	+1.5541 [+1.2463, +2.0117]	+1.5584 [+1.2001, +2.0313]
AP large	+0.0000 [+0.0000, +0.0000]	+0.0000 [+0.0000, +0.0000]
Free refinement (640) − YOLO26s-seg (640)		
AP	+1.1932 [+0.7921, +1.6278]	+1.4832 [+1.1247, +1.9148]
AP50	-0.6586 [-1.1240, -0.0974]	-0.7032 [-1.0035, +0.2869]
AP75	+1.2026 [+0.0502, +2.2862]	+0.9058 [-0.0607, +1.6786]
AP90	+3.9269 [+2.3497, +5.2729]	+3.2133 [+2.0573, +5.0895]
AP small	+0.8808 [-0.6487, +2.2774]	+1.7656 [+0.1297, +3.1758]
AP medium	+1.9075 [+1.4123, +2.3803]	+2.2508 [+1.7558, +2.7417]
AP large	+0.0000 [+0.0000, +0.0000]	+0.0000 [+0.0000, +0.0000]
Native-area − free decoding		
AP	+0.1225 [-0.3317, +0.4744]	-0.1778 [-0.5356, +0.3120]
AP50	+0.8563 [+0.1392, +1.1866]	+0.7032 [-0.2869, +1.0035]
AP75	-0.5683 [-1.2131, +0.3967]	-0.5659 [-1.1274, +0.7508]
AP90	-1.0301 [-2.3757, +0.2025]	-0.1672 [-1.9483, +0.9276]
AP small	+1.1590 [-0.3512, +2.7068]	+0.8438 [-0.5848, +2.6303]
AP medium	-0.3534 [-0.7305, +0.1918]	-0.6924 [-1.0887, -0.1746]
AP large	+0.0000 [+0.0000, +0.0000]	+0.0000 [+0.0000, +0.0000]
Direct − residual prediction, both with native-area decoding		
AP	+0.0638 [-0.0612, +0.2434]	-0.0544 [-0.1519, +0.1761]
AP50	+0.0000 [-0.1145, +0.0354]	+0.4731 [+0.0000, +0.6143]
AP75	-0.0535 [-0.4418, +0.6034]	-0.5731 [-0.9100, +0.2080]
AP90	+0.1161 [-0.7094, +1.0029]	+0.2007 [-0.3836, +0.9062]
AP small	-0.2717 [-0.6445, +0.1353]	-0.1278 [-0.6009, +0.3257]
AP medium	+0.2216 [+0.0311, +0.5877]	+0.1340 [-0.0996, +0.3289]
AP large	+0.0000 [+0.0000, +0.0000]	+0.0000 [+0.0000, +0.0000]
(b) Complete-system quality references		
ProtoRefine (640) − YOLO26s-seg (896)		
AP	-0.7796 [-1.2417, -0.3828]	-1.0836 [-1.4144, -0.5526]
AP50	-0.3676 [-0.9046, +0.0262]	-0.3728 [-1.1587, -0.0333]
AP75	-1.4442 [-2.4186, -0.4765]	-1.9362 [-2.7650, -0.8472]
AP90	-0.1777 [-1.3897, +1.0004]	-0.2375 [-2.1342, +1.0604]
AP small	-2.7529 [-4.1777, -1.3245]	-3.2121 [-4.4843, -1.8149]
AP medium	-0.3022 [-0.7812, +0.2034]	-0.2296 [-0.7219, +0.2652]
AP large	-0.2808 [-0.6737, +0.0554]	-0.4827 [-0.8668, -0.2171]

Table S3 Paired test differences (pp; 95% image-bootstrap intervals). Each column uses its own YOLO26s-seg model. External baselines are shared; signs denote first method minus second.

Continued on the next page

Metric	Main YOLO26s-seg model Difference (pp) [95% interval]	Second YOLO26s-seg model Difference (pp) [95% interval]
ProtoRefine (640) − YOLO11s-seg (640)		
AP	+1.7588 [+1.3562, +2.0381]	+1.6562 [+1.2938, +2.0451]
AP50	+0.5559 [-0.1173, +0.8758]	+0.6015 [-0.2559, +0.9193]
AP75	+1.5241 [+0.4912, +2.5642]	+1.6341 [+0.7474, +2.6437]
AP90	+3.7737 [+2.0621, +4.7939]	+3.7909 [+1.8655, +4.9272]
AP small	+3.3005 [+2.0647, +4.4814]	+3.6073 [+2.3117, +4.9052]
AP medium	+2.1079 [+1.5729, +2.5711]	+1.9754 [+1.4012, +2.5373]
AP large	-0.1783 [-0.4002, +0.0581]	-0.1246 [-0.3904, +0.0860]
ProtoRefine (640) − RTMDet-Ins-tiny (640)		
AP	+3.9334 [+3.3947, +4.4748]	+3.8308 [+3.3468, +4.4599]
AP50	+1.9236 [+1.0311, +2.4835]	+1.9692 [+0.9747, +2.5130]
AP75	+3.4917 [+2.4194, +4.9218]	+3.6016 [+2.3239, +5.0302]
AP90	+5.0894 [+3.2522, +6.8873]	+5.1066 [+3.2733, +7.0243]
AP small	+10.5310 [+8.6216, +12.5322]	+10.8378 [+8.9719, +12.8157]
AP medium	+3.2929 [+2.6579, +3.9820]	+3.1604 [+2.4718, +3.9014]
AP large	+0.3170 [+0.0132, +0.6508]	+0.3707 [+0.0090, +0.6784]
ProtoRefine (640) − RTMDet-Ins-tiny (896)		
AP	+0.4350 [-0.1168, +0.8088]	+0.3324 [-0.1575, +0.8596]
AP50	+0.6505 [-0.1374, +0.9682]	+0.6961 [-0.2152, +1.0221]
AP75	+0.1435 [-0.8467, +1.4404]	+0.2535 [-0.6580, +1.5290]
AP90	+0.7477 [-0.9915, +2.0235]	+0.7649 [-1.0283, +2.0516]
AP small	+1.7395 [+0.0028, +3.4688]	+2.0463 [+0.3243, +3.8232]
AP medium	+0.2077 [-0.2982, +1.0378]	+0.0752 [-0.4386, +0.9347]
AP large	-0.3217 [-0.6329, -0.0165]	-0.2680 [-0.5671, -0.0311]

Table S3 Paired test differences in pp with 95% intervals (continued).

4 Sensitivity to head initialization and local support

4.1 Head initialization

We repeat the complete coarse/full input and free/native-area decoding comparison at local-head initialization seeds 7, 42, and 123, keeping the detector and epoch sample orders fixed. Figure S2 visualizes the incremental value of explicit instance factors and its interaction with decoding. These experiments assess local-head initialization sensitivity, not independent detector-training variability.

For metric Q, the interaction is $I_s = (Q_{\text{Full,Area}} - Q_{\text{Coarse,Area}}) - (Q_{\text{Full,Free}} - Q_{\text{Coarse,Free}})$. A negative value means that explicitly exposing the factors contributes less under native-area decoding. Full inputs improve free-decoding AP in all three runs, while their native-area advantage changes sign. The AP interaction is negative throughout, with mean -0.4449 pp and sample standard deviation 0.0414 pp. The corresponding input gains average 0.5270 ± 0.0999 pp under free decoding and 0.0821 ± 0.1358 pp under native-area decoding (mean $\pm$ sample standard deviation). AP90 does not show the same consistent interaction direction.

This replication supports a conditional AP observation rather than a unique calibration explanation or a significance claim. The heads have different widths and initialization draws, and only the local heads are retrained. The complete comparison is in Table S4.

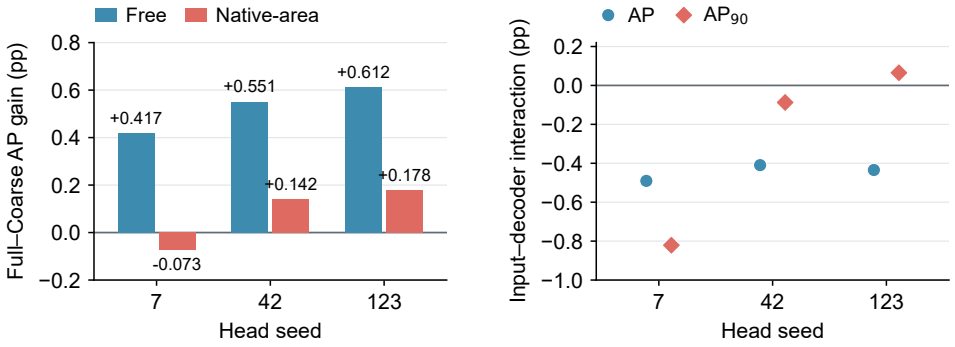
(a) Complete mask metrics

Input / decoding	AP	AP50	AP75	AP90	AP small	AP medium	AP large
Head seed 7							
Coarse / free	83.3642	96.7102	89.0290	67.5270	47.2790	88.1707	98.5467
Full / free	83.7815	96.8982	90.1362	67.4158	49.3671	88.1370	98.5467
Coarse / native-area	84.3780	97.6277	90.5625	68.3359	52.0063	88.5066	98.5467
Full / native-area	84.3047	97.6297	90.4904	67.4037	52.0751	88.2620	98.5467
Head seed 42 (main setting)							
Coarse / free	83.3828	96.7289	89.0180	67.3781	47.2848	88.0706	98.5467
Full / free	83.9342	96.6397	90.1751	67.4457	49.5083	88.5078	98.5467
Coarse / native-area	84.3820	97.6118	90.5129	68.4222	51.8755	88.6971	98.5467
Full / native-area	84.5238	97.6434	90.4975	68.4022	52.4034	88.6743	98.5467
Head seed 123							
Coarse / free	83.3192	96.7289	88.9343	67.4972	47.3892	87.9678	98.5467
Full / free	83.9317	96.6763	90.2610	67.4883	49.5383	88.3540	98.5467
Coarse / native-area	84.3819	97.6118	90.2145	68.3691	51.7426	88.6529	98.5467
Full / native-area	84.5599	97.5369	90.5391	68.4253	52.2541	88.5234	98.5467

(b) Conditional input gains and interaction

Head seed	Full–Coarse Free AP	Full–Coarse Area AP	I_{AP}	I_{AP90}
7	+0.4173	-0.0733	-0.4905	-0.8210
42	+0.5513	+0.1418	-0.4096	-0.0875
123	+0.6125	+0.1779	-0.4346	+0.0650

Table S4 Input–decoding comparison over three head initializations on the same 265 development images. Panel (b) reports differences in pp; I is the area-decoding input gain minus the free-decoding input gain. The full and coarse heads retain their original widths.



(a) (b)

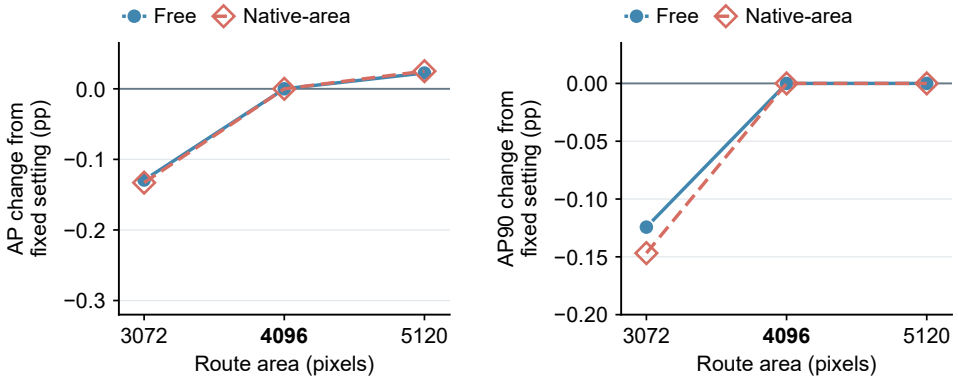
Fig. S2 Decoder-dependent value of explicit instance factors across three local-head initializations. (a) Full-minus-coarse mask AP gains under free thresholding and native-area reconstruction. (b) Input–decoder interactions for AP and AP90, defined as the native-area input gain minus the free-decoding input gain. The AP interaction is negative for all three seeds, whereas the AP90 interaction changes sign. All experiments share the same fixed segmentation model and development data; complete results are reported in Table S4.

4.2 Routing area and support size

Routing controls which detections are refined; the support factor controls the surrounding spatial context. Table S5 varies one at a time. AP changes little over the tested routing areas, but increasing support to 1.40 reduces native-area AP90 by 0.97 pp. The curves in Figures S3–S4 therefore show a local tradeoff rather than uniform improvement from more context.

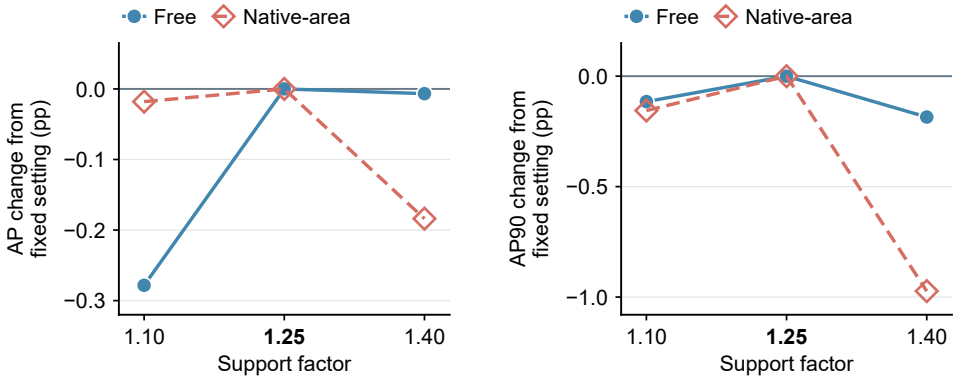
Route area / support	Routed candidates	Free AP	Free AP90	Area AP	Area AP90
Fixed setting					
4096 / 1.25	1179	83.9342	67.4457	84.5238	68.4022
Route-area variation					
3072 / 1.25	1067	83.8049	67.3214	84.3908	68.2554
5120 / 1.25	1298	83.9566	67.4457	84.5488	68.4022
Support-factor variation					
4096 / 1.10	1179	83.6559	67.3306	84.5058	68.2465
4096 / 1.40	1179	83.9275	67.2607	84.3399	67.4291

Table S5 Local routing and support variations around the main setting (4096 / 1.25). Area denotes native-area decoding.



(a) (b)

Fig. S3 Local route sensitivity. Changes in (a) AP and (b) AP90 relative to route area 4096, with support fixed at 1.25. Only observed settings are connected; the bold tick marks the fixed setting.



(a) (b)

Fig. S4 Local support sensitivity. Changes in (a) AP and (b) AP90 relative to support 1.25, with route area fixed at 4096. Only observed settings are connected; the bold tick marks the fixed setting.

5 Additional test comparisons

The second YOLO26s-seg model shows the same distinction between improving native masks and choosing a decoding rule. Both refinements improve on native 640-pixel prediction, but free decoding has slightly higher AP than native-area decoding (Figure S5). The area rule is therefore not uniformly better across the two fixed models.

Figure S6 places the original official-test systems in the AP–AP90 plane. Free prediction and native-area decoding emphasize different aspects of mask quality, while

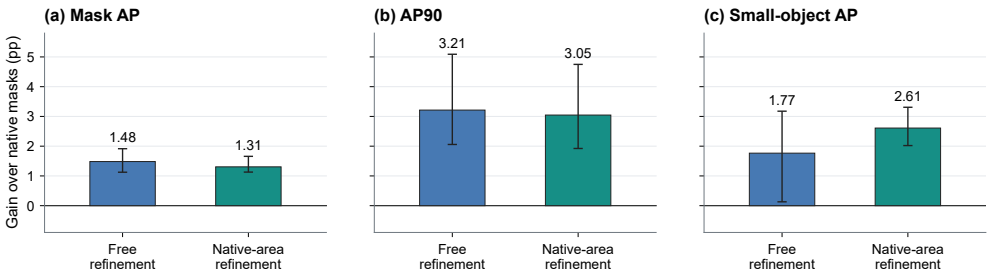


Fig. S5 Refinement on the second detector. Test gains over its native 640-pixel prediction for free and native-area decoding, with 95% paired-image intervals.

native 896-pixel inference remains a strong reference. These comparisons motivate reporting strict overlap and small-object quality alongside overall AP.

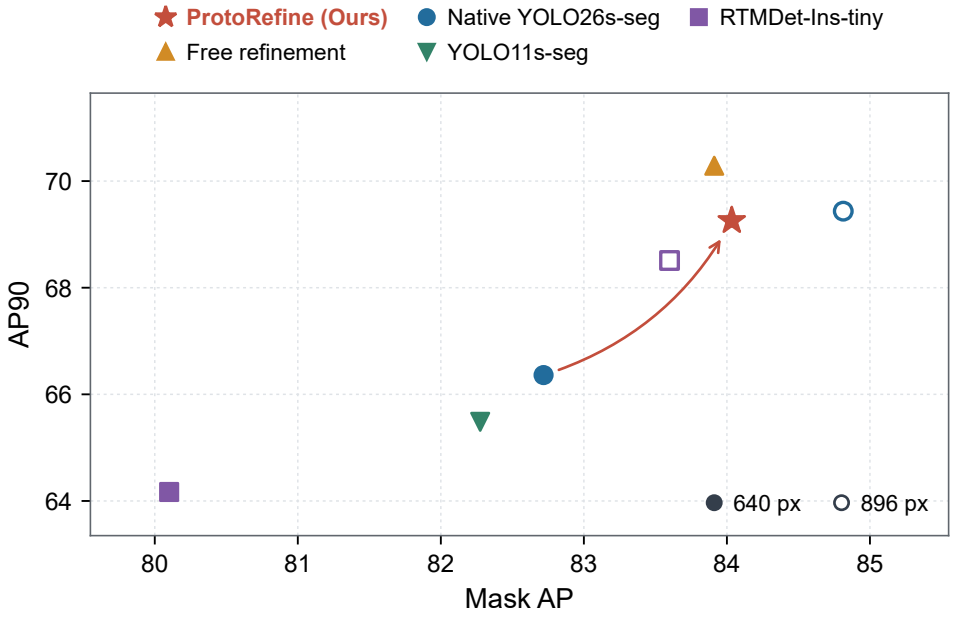


Fig. S6 Official-test quality across systems. Mask AP and AP90 from Table S1; filled and open markers denote 640- and 896-pixel inputs, respectively. The arrow connects native YOLO26s-seg at 640 pixels to ProtoRefine (Ours).

6 Refinement alternatives and geometric behavior

6.1 Training and control definitions

Shared training. Local heads use cached fit instances. Candidates are positive at best box IoU ≥ 0.5, negative below 0.1 and ignored in between. Mean binary cross-entropy is applied to every training grid; positive instances also receive a soft Dice term with unit smoothing and equal weight. Training uses AdamW, learning rate 10^{-3} with cosine decay to 10^{-4}, weight decay 10^{-4}, AMP, instance batch size 32 and 12 epochs. The final epoch is used without EMA or development-based selection.

Prediction and decoding. Free decoding thresholds the rendered field at zero. Native-area decoding instead retains the foreground count of the native mask. The residual alternative predicts $z_i + 3\tanh(f_\theta(X_i))$ and uses the same area decoder. Its initial function and output range differ from direct prediction, so the test comparison in Table S3 concerns the two formulations rather than an isolated skip-connection effect.

Input information and capacity. Coarse controls receive logits and coordinates only. The masked-input version retains the nominal 25,144-parameter head but has

8440 parameters connected to active inputs. The wider $3 \to 51 \to 51 \to 1$ head has 24,940 parameters, 0.81% fewer than the full head. Width 51 is determined by this budget. Fit instances, sample order, optimizer and duration are shared; different widths also change initialization and optimization trajectories. This comparison tests explicit access to prototypes and coefficients, whose composition already contributes to the coarse field.

Point-based refinement. The PointRend-inspired adaptation uses 25,442 parameters and 196 training points per instance: 147 uncertain points selected from 588 uniform candidates and 49 fresh uniform points. Three 96-unit hidden layers condition on a coarse 7×7 field. Inference subdivides $7 \to 14 \to 28 \to 56 \to 112 \to 224$ using 3332 predictions. Its point supervision and rendering differ from the dense head's 4096 training pixels; comparable parameter counts do not imply equal computational cost.

Contrast	ΔAP [95% CI]	ΔAP90 [95% CI]
Free dense − Point	+0.0142 [-0.1803, +0.2774]	-0.9324 [-1.2006, +0.7852]
ProtoRefine − Point	+0.6038 [+0.2457, +0.9184]	+0.0242 [-0.9656, +1.1498]

Table S6 Paired development comparisons with the prototype-based point-refinement adaptation. Differences are in pp, with 95% paired-image intervals from 1000 common draws.

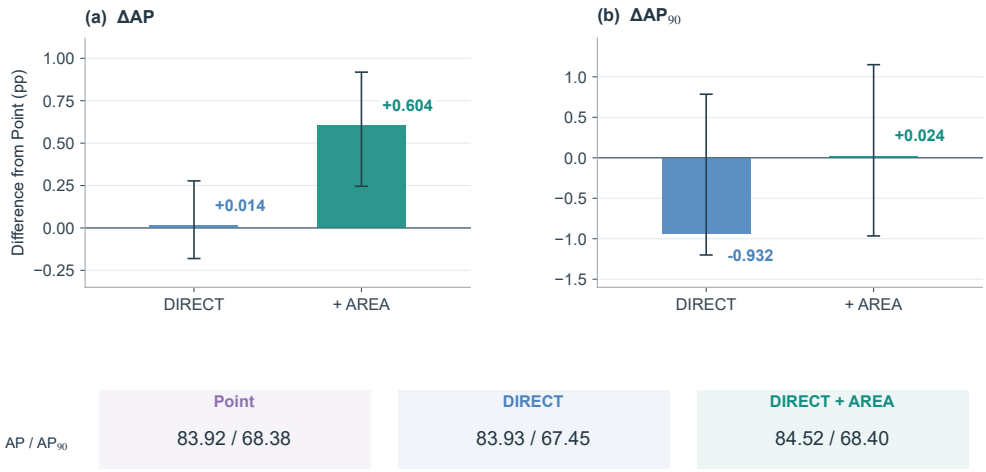


Fig. S7 Dense and point-based refinement. Development AP and AP90 differences from the specified point adaptation, with 95% paired-image intervals.

Free dense prediction and the point adaptation have nearly identical development AP. Adding native-area decoding yields a 0.60 pp AP advantage over the point adaptation, while the AP90 interval still spans zero (Table S6). The result supports comparing the complete refinement pipelines; it does not establish that dense prediction alone is superior.

6.2 Foreground quantity and contour errors

To separate contour correction from detection changes, the geometry analysis holds native-box correspondences fixed at box IoU ≥ 0.5, including low-score candidates. The main and second models have 1458 and 1459 matched ground-truth instances, respectively, with 864 routed matches each. The model-specific sets need not contain identical instances. Table S7 measures overlap, boundary alignment and foreground quantity on these routed matches.

Mean improvement does not imply an instance-wise advantage. Native-area decoding improves average IoU and boundary F1 over native masks. Against free decoding, however, slightly more routed instances lose IoU than gain it, despite the

positive mean change (Table S7(b)). The count constraint redistributes errors: fixing foreground quantity can recover missing pixels while retaining spill elsewhere. Equal omission and spill changes relative to native masks follow from the fixed-area identity.

(a) Conditional means

Method	Empty	IoU	F1	Omission	Spill	Area error
Main YOLO26s-seg model						
Native (640)	0	86.1469	87.9549	5.9819	10.2012	4.2193
Free refinement	5	85.7152	88.4135	9.6688	5.7967	-3.8721
ProtoRefine	0	87.2782	89.7686	5.2942	9.5135	4.2193
Residual + area	0	87.2738	89.6972	5.3044	9.5237	4.2193
Fit-calibrated bias	9	85.0690	87.5402	10.9502	5.0035	-5.9467
Point adaptation	6	85.8122	88.3796	8.8474	6.7507	-2.0967
Second YOLO26s-seg model						
Native (640)	0	86.0349	88.2489	6.1047	10.2642	4.1595
Free refinement	9	85.6905	88.6393	9.4843	6.0695	-3.4148
ProtoRefine	0	87.3152	90.0104	5.3080	9.4675	4.1595
Residual + area	0	87.3280	90.0359	5.3094	9.4689	4.1595
Fit-calibrated bias	11	84.9334	87.7571	11.0593	5.0271	-6.0322

(b) Native-area minus comparator

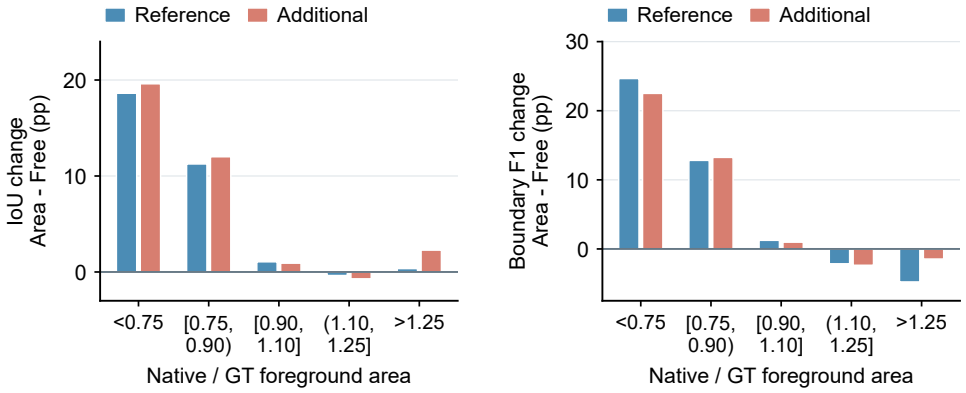
Comparator	ΔIoU	ΔF1	Improved	Degraded	Unchanged
Main YOLO26s-seg model					
Native (640)	+1.1313	+1.8137	639	190	35
Free refinement	+1.5630	+1.3551	421	443	0
Second YOLO26s-seg model					
Native (640)	+1.2803	+1.7615	634	196	34
Free refinement	+1.6247	+1.3711	411	453	0

Table S7 Geometry on 864 routed native-box matches per model. (a) Conditional means; omission is FN/GT area, spill is FP/GT area and signed area error is (prediction–GT)/GT area, all in percent. F1 uses a two-pixel boundary tolerance. Empty counts are numbers of empty masks. (b) Mean changes in pp and counts of individual IoU outcomes for ProtoRefine minus its comparator.

Native/GT ratio	n	Area–native ΔIoU	Area–native ΔF1	Area–free ΔIoU	Area–free ΔF1
Main YOLO26s-seg model					
< 0.75	12	+1.9482	+0.9577	+18.6226	+24.6615
$[0.75, 0.90)$	40	+2.3034	+1.8549	+11.2545	+12.8177
$[0.90, 1.10]$	658	+0.9905	+1.6943	+1.0611	+1.2489
$(1.10, 1.25]$	105	+1.2962	+2.2015	-0.3696	-2.1498
> 1.25	49	+1.5110	+2.7622	+0.3554	-4.7736
Second YOLO26s-seg model					
< 0.75	9	+0.2397	-5.8226	+19.6086	+22.5118
$[0.75, 0.90)$	49	+2.2750	+0.8934	+12.0064	+13.2492
$[0.90, 1.10]$	656	+1.1682	+1.7528	+0.9183	+0.9819
$(1.10, 1.25]$	102	+1.5116	+2.7639	-0.7141	-2.3639
> 1.25	48	+1.5005	+2.0590	+2.2780	-1.4617

Table S8 Routed-match geometry by native-to-ground-truth foreground area ratio. Area denotes native-area decoding. Differences are in pp; both comparisons use the same native-defined strata. Ground truth is used only for this analysis.

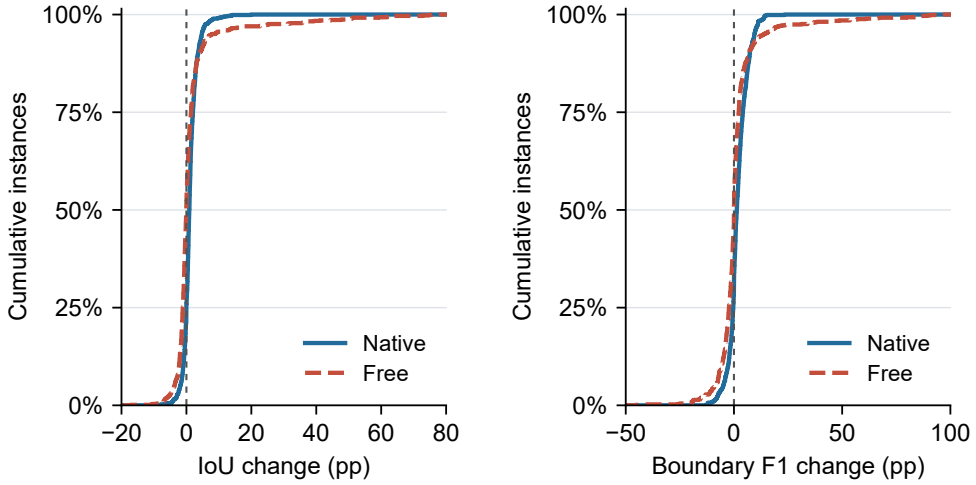
An oversized native mask limits boundary correction. In both models, the strata where native foreground exceeds GT by more than 10% have lower boundary F1 under native-area than free decoding (Table S8; Figure S8). Preserving area can stabilize foreground quantity, but it also preserves errors in that quantity. This observation identifies a failure condition of the constraint, rather than evidence for uniform improvement across mask shapes.



(a) (b)

Fig. S8 Geometry by native foreground area. Native-area minus free decoding in (a) mean IoU and (b) boundary F1. Reference/Additional are the main/second models; stratum counts are in Table S8.

The complete distributions in Figure S9 expose the tails behind the main figure's central views. Gains on a subset of masks coexist with losses or ties in others.



(a) Mask IoU (b) Boundary F1

Fig. S9 Complete distributions of instance-wise geometric changes. ProtoRefine minus native prediction or free decoding on the same 864 routed development matches used in main Figure 7. Both curves include every matched instance, including the negative and positive tails. The vertical line marks no change; boundary F1 uses a two-original-pixel tolerance. The main figure magnifies the central horizontal ranges without changing the cumulative denominator.

6.3 Properties of foreground-count-constrained reconstruction

The following properties describe the existing decoder. They explain its output constraint without adding a training objective or an assumption that the native count is correct. Let S_i be the rendered support, $\Omega_i = S_i \cup N_i$, and $K_i = |N_i|$; all sets use the same original-image pixel grid.

Selection at a fixed count. For a finite score field s_i on Ω_i, consider

$$M_i^* \in \underset{M \subseteq \Omega_i,\ |M| = K_i}{\arg\max} \sum_{x \in M} s_i(x). \tag{S1}$$

If a selected pixel scores below an unselected pixel, exchanging them increases the objective. Repeating this exchange yields a set of the K_i highest scores. Ties can yield multiple maximizers; original-image row-major order specifies the implementation's

choice. This elementary selection problem contains no boundary or connectivity term and optimizes predicted scores, not ground-truth overlap.

In the implementation, scores are finite on S_i and negative infinity on $N_i \setminus S_i$. If $0 < K_i \leq |S_i|$, all selected pixels can come from the support. If $K_i > |S_i|$, all support pixels are selected and the remaining positions are taken from $N_i \setminus S_i$ in row-major order. Capacity is sufficient because $N_i \subseteq \Omega_i$. For this latter case, a direct sum of extended-real scores would be negative infinity for every feasible set and would not characterize the selection. An equivalent finite-score formulation assigns all exterior pixels any common value below the minimum support score, then applies Equation S1 with the same tie rule. This is an interpretation of the existing ordering, not a change to its scores. An empty native mask gives an empty output.

Ranking and thresholding. If $0 < K_i < |\Omega_i|$ and the cutoff separates distinct finite scores, a threshold between the K_ith and $(K_i + 1)$th scores reproduces the selected set. With tied scores, a scalar threshold alone may not produce exactly K_i pixels. For fixed Ω_i and K_i, any strictly increasing transformation of the rendered finite scores that retains the exterior ordering preserves the output. In particular, adding a common bias leaves native-area selection unchanged but can alter free zero-threshold decoding. These are exact-order properties; finite-precision transformations that introduce ties require the stated tie rule. The predictor itself is trained with BCE and Dice, so ranking-only inference does not imply ranking-only learning.

Error conservation and attainable overlap. Let G_i be a nonempty annotation with $g_i = |G_i|$. For any prediction A, $|A| = TP(A)+FP(A)$ and $g_i = TP(A)+FN(A)$, hence

$$FP(A) - FN(A) = |A| - g_i. \tag{S2}$$

Applying this identity to M_i and N_i gives equal changes in FP and FN. Division by the same g_i preserves the equality for omission and spill, as does averaging over the same matched instances. The equal reductions in Table S7 are therefore a consequence of fixed count; the favorable direction of those changes is an empirical result.

Writing $t_i = |M_i \cap G_i|$, fixed count also gives

$$\mathrm{IoU}(M_i, G_i) = \frac{t_i}{K_i + g_i - t_i} \leq \frac{\min(K_i, g_i)}{\max(K_i, g_i)}. \tag{S3}$$

The expression increases with t_i and $t_i \leq \min(K_i, g_i)$. Equality requires the smaller mask to be contained in the larger; the admissible domain can prevent this. Equivalently, $FP \geq \max(K_i - g_i, 0)$ and $FN \geq \max(g_i - K_i, 0)$. Thus an overestimated count necessarily retains some false foreground, while an underestimated count necessarily misses some annotated foreground. These bounds do not imply that free decoding attains a better overlap, nor do they determine boundary F1: boundary quality depends on the spatial arrangement of errors and the evaluation tolerance. Ground truth enters only this analysis, not inference.

7 Data and evaluation details

7.1 Datasets and data roles

StrawDI Db1 contains 3100 annotated images with official training, validation and test roles of 2800, 100 and 200 images (Pérez-Borrero et al. 2020). Our internal development uses a fixed 2427-image fit role and a 265-image development role containing 1471 instances. These internal roles are not the official validation split. Final strawberry evaluation uses all 200 official test images and 1132 annotated instances. The source paper describes collection across 20 plantations; the available metadata do not certify plantation-independent experimental splits.

The internal split is constructed only from the official training role. Numeric filename indices are grouped into 50-ID blocks, and six blocks are assigned to development using seed 20261001. A remaining image is excluded from training when its numeric index lies within 10 of any development-image index. This buffer excludes 108 images,

so the 2800 source images are partitioned into 2427 fit, 265 development and 108 buffer-excluded images. Filename order is an unverified proxy for acquisition or scene order: the buffer reduces immediate-neighbor overlap but does not establish scene-independent partitions. The official 100-image validation role had previously informed problem selection and is not treated as a fresh confirmatory sample.

Test models, comparisons and the statistical procedure were fixed before test-content access. The later development controls did not alter those test models.

7.2 Models and comparisons

Both YOLO26s-seg models have 11,434,463 parameters, 32 prototypes and one class, using Ultralytics 8.4.173 (Jocher et al. 2026). The main model is used for the manuscript analyses; the second model supplies an additional within-architecture comparison. Each refinement head is trained on its own fixed detector.

The test comparison includes native predictions at 640 and 896 pixels, free and native-area refinement, and residual prediction with native-area decoding for each YOLO26 model. YOLO11s-seg uses the same fit split, batch size eight, seed 42 and final 60-epoch EMA model; RTMDet-Ins-tiny uses its same-fit epoch-60 model. YOLO inference uses score threshold 0.001, box NMS IoU 0.7 and at most 300 candidates. RTMDet-Ins-tiny uses score threshold 0.05, NMS IoU 0.6, 1000 pre-NMS candidates, at most 100 masks and binary-mask threshold 0.5. Each family retains its own inference procedure.

7.3 Evaluation and uncertainty

We report COCO mask AP, AP50, AP75, AP90 and small-, medium- and large-object AP on a 0–100 scale. Differences are percentage points (pp). Native-area refinement versus each model's own native 640-pixel prediction is a primary test comparison; all 16 paired comparisons were specified before test-content access.

Test intervals use 1000 paired image-bootstrap draws, with common sampled image identities across methods and global COCO accumulation recomputed for each draw. Percentile 95% intervals are conditional on the fixed models; they do not include training variance or adjustment for multiple comparisons.

The point-refinement and coarse-input controls use the development split. The capacity-matched head and point-comparison intervals were added after the final test evaluation. The latter use 1000 paired draws with seed 20261012 for AP and AP90. The original capacity-control seed 42 is supplemented by the two head initializations in Section 4.1; their means, ranges and sample standard deviations are descriptive, without new bootstrap draws or a significance test.

7.4 Matched-instance analysis

Mask IoU and boundary F1 are computed on fixed native-box matches, with boundary tolerance two pixels in the original image. Omission is FN area divided by GT foreground area; spill is FP area divided by GT foreground area. Signed area error is $(|M| - |G|)/|G|$. Means describe routed matched instances and exclude missed or unmatched objects, whereas whole-image AP includes detection and ranking effects.

7.5 Computational scope and latency

Table S9 reports batch-one end-to-end latency on an NVIDIA GeForce RTX 4060 Laptop GPU with 8 GB memory. All systems use the original fixed YOLO26 checkpoint and FP32 prediction, with cuDNN TF32 enabled and matrix-multiplication TF32 disabled; ProtoRefine uses the final seed-42 head. We select 32 fit images at evenly spaced positions in the image-ID-sorted training list, without using image content or model outcomes. Each system processes all 32 images in each of the six possible system orders, giving 192 observations per system and 576 in total. Each system block is warmed up with eight predictions before timing.

System	Input	Mean	Median	P95	Peak tensor memory
Native YOLO26s-seg	640	96.46	79.99	160.96	284.70
ProtoRefine	640	131.25	102.78	246.02	284.82
Native YOLO26s-seg	896	96.37	84.50	156.29	270.40

Table S9 Batch-one end-to-end time to output RLE on 32 fixed fit images. Times are in ms and peak allocated tensor memory is in MiB. Each system has 192 observations across six execution orders.

ProtoRefine adds 34.79 ms per image (36.1%) relative to native 640-pixel inference, while its peak allocated tensor memory remains close to that of the native model. The head is compact in parameters, but its rendering, transfer and rank-based reconstruction contribute a measurable end-to-end cost. Native 896-pixel inference averages 96.37 ms in this workload, below ProtoRefine; these results do not establish refinement as the cheaper alternative to increasing resolution.

Timing begins before image-file decoding and ends after final original-image RLE masks, with CUDA synchronization at both boundaries. It includes preprocessing, model inference, native-mask rendering, candidate and mask transfer to CPU, and any local prediction and rank-based selection. Model loading, warmup and result checks are outside the timed span. Mean, median and P95 use all 192 observations per system; P95 is the linearly interpolated 95th percentile. Peak allocated tensor memory includes the process's persistent model tensors, but excludes CPU memory and is not whole-device VRAM usage.

Measurements use the same Windows WDDM desktop session, plugged-in power setting, four main Torch threads and PyTorch 1.13.1. No other training or inference jobs run concurrently; the remote-control proxy remains active. Its influence is not separately quantified, so these measurements characterize this desktop environment. Across the six orders, block-mean times range from 95.16 to 97.68 ms for native 640, 126.26 to 136.00 ms for ProtoRefine and 94.71 to 98.41 ms for native 896. The native settings retain their own outputs, averaging 21.03 candidates per image at 640 and 18.28 at 896; their similar total times do not isolate the effect of input resolution.

8 Complete-system baseline settings

Mask R-CNN R50-FPN (He et al. 2017), RTMDet-Ins-s (Lyu et al. 2022) and YOLACT–ResNet50-FPN (Bolya et al. 2019) provide region-based, dynamic-mask and prototype-based complete-system comparisons on the development split. Each uses a predetermined 60-epoch recipe and final model, without development-based checkpoint selection.

8.1 Shared data and initialization

All three use the same 2427 fit images (14,123 instances) and 265 development images (1471 instances). Initialization uses the official OpenMMLab COCO instance-segmentation models: Mask R-CNN R50-FPN 2x, RTMDet-Ins-s 300e and YOLACT R50-FPN. Compatible pretrained weights are retained; classification and auxiliary semantic outputs are adapted to one class.

8.2 Architecture-specific training recipes

Mask R-CNN. The model retains ResNet50-FPN, RPN, RoIAlign, the box head and the four-convolution mask head with 28×28 targets. The first-stage backbone freezing and batch-normalization evaluation policy are retained. Training uses aspect-preserving resizing within a 640×640 envelope, horizontal flip probability 0.5 and divisor-32 padding. AMP and four-microbatch accumulation give nominal optimizer batch eight, without dropping the final smaller microbatch.

Setting	Mask R-CNN R50-FPN	RTMDet-Ins-s	YOLACT R50-FPN
Model and training setup			
Total parameters	43,971,158	10,152,738	34,727,123
Fit epochs / seed	60 / 42	60 / 42	60 / 42
Microbatch / accumulation	2 / 4	8 / 1	2 / 4
Arithmetic	AMP	AMP	FP32
Optimization			
Optimizer	SGD; momentum 0.9	AdamW	SGD; momentum 0.9
Initial learning rate	0.01	0.0001	0.001
Weight decay	0.0001	0.05	0.0005
Schedule	Decay by 0.1 at 40, 55	Cosine to 0.000001	Decay by 0.1 at 22, 46, 54, 57
Warmup	500 microsteps; factor 0.001	100 steps; factor 0.1	500 microsteps; factor 0.1
Final model			
Model used	Last epoch; no EMA	Last EMA; momentum 0.0002	Last epoch; no EMA
Inference			
Score / max masks	0.05 / 100	0.05 / 100	0.05 / 100
NMS	Box IoU 0.5	Box IoU 0.6	Fast NMS IoU 0.5
Binary-mask threshold	0.5	0.5	0.5

Table S10 Training and inference settings for the three additional baselines. All use the same fit/development split. Optimizer batch size is eight; YOLACT batch normalization uses two images.

RTMDet-Ins-s. The model retains the official small architecture (depth 0.33, width 0.5, neck/head width 128), native dynamic-mask losses and the augmentation pipeline used by RTMDet-Ins-tiny. The second-stage pipeline starts at epoch 51. AdamW applies zero weight decay to normalization parameters and biases; inference uses the epoch-60 EMA.

YOLACT. The model retains ResNet50-FPN, prototype linear assembly, native OHEM classification, box, mask and auxiliary-semantic losses, Expand, MinIoURandomCrop and photometric augmentation. Its declared adaptation changes the original 550-pixel/55-epoch recipe to 640-pixel distorted resize and 60 epochs. The FPN grids are 80/40/20/10/5, with the original anchor construction. FP32 microbatch two and four-step accumulation give nominal optimizer batch eight, while batch normalization actually sees two images (`norm_eval=False`). YOLACT uses the SGD schedule in Table S10, without learning-rate scaling, AMP or EMA. Fast NMS retains `nms_pre`=1000 and `top_k`=200. No recipe was adjusted in response to development performance.

9 Architecture and tensor organization

The backbone, neck and native segmentation path are fixed during local-head training (main Figures 1–2). At square 640-pixel input, the neck supplies 128×80×80, 256×40×40 and 512×20×20 feature tensors. Proto26 produces 32×160×160 shared spatial prototypes.

9.1 Local representation and prediction

Prototype features and the coarse field are bilinearly sampled on the same 64×64 pixel-center grid, with border padding and `align_corners=False`. Broadcasting the 32 instance coefficients and adding two normalized coordinate maps gives $32+32+2+1 = 67$ input channels. RGB is not an additional head input.

Two 3×3 convolutions (67-to-29 and 29-to-29) with ReLU precede a 1×1 projection to one logit, followed by scaling by three. Resolution stays 64×64. Including biases, the head has $(67 \times 29 \times 9 + 29) + (29 \times 29 \times 9 + 29) + (29 + 1) = 25{,}144$ parameters.

9.2 Primary routing and local-head training recipe

Setting	Value
Routing and local support	
Native input setting	640 pixels
Refinement routing	Native score ≥ 0.05; positive box area ≤ 4096 pixels at the 640-pixel scale
Routing limit	At most 16 candidates per image, in descending native-score order with stable ties
Local support	Predicted box expanded by 1.25 and clipped to the image
Local prediction grid	64×64 pixel centers
Head training	
Head training	12 epochs; seed 42; instance batch size 32; AMP
Optimizer	AdamW; weight decay 10^{-4}
Learning-rate schedule	Cosine decay from 10^{-3} to 10^{-4}
Model used	Last epoch; no EMA or development-based selection

Table S11 Routing and training settings for ProtoRefine. The native detector is fixed during head training.

10 Development results and qualitative examples

10.1 Complete-system comparison

The region-based, dynamic-mask and prototype-based systems are compared on the same 265 development images (Table S12). Their recipes are detailed in Section 8. These are already-seen development data, and cross-family differences combine architecture and training effects. The refinement gain is assessed against the native YOLO26s-seg model from which ProtoRefine is constructed.

System	AP	AP50	AP75	AP90	AP_S	AP_M	AP_L
Other complete systems							
YOLACT–ResNet50-FPN	76.3700	94.6708	85.0983	52.7165	38.2770	81.6619	96.6678
Mask R-CNN R50-FPN	78.5759	92.9148	85.5925	61.3678	37.7288	84.2311	97.3723
RTMDet-Ins-tiny	80.5593	95.9868	85.8503	61.3619	42.2582	85.4378	98.3528
RTMDet-Ins-s	81.5585	95.8772	86.8681	63.2694	44.0616	86.4128	98.5215
YOLO11s-seg	83.0289	97.4098	89.5162	64.4270	50.3182	86.6378	98.4286
Native system and its refinement							
Native YOLO26s-seg	83.1275	97.3186	89.6280	64.3380	50.0880	86.7528	**98.5467**
ProtoRefine (Ours)	**84.5238**	**97.6434**	**90.4975**	**68.4022**	**52.4034**	**88.6743**	**98.5467**

Table S12 Complete-system mask quality on the shared development split (0–100 scale). Bold marks the highest score, including ties. These results are separate from the official test comparison.

The clearer separation is on small objects. Large-object AP changes little across the stronger systems, while small-object AP distinguishes their masks more clearly (Figure S10). ProtoRefine improves its own native small-object AP by 2.32 pp; its large-object score is unchanged. The full-split comparison provides the aggregate context for the contour examples below.

10.2 Visible-contour examples

Main Figure 5 illustrates favorable contour corrections; Figures S11 and S12 add different source images showing failures and little-change outcomes. They expose cases where inheriting the native count retains excess foreground, where refinement disturbs an accurate outline, and where little correction is needed.

Selection and display. The main-paper examples were selected for ProtoRefine-to-GT agreement and visible comparator errors, with target IoU and boundary F1 ≥ 0.90, score ≥ 0.5, GT area ≥ 400 pixels and native-relative IoU gain ≥ 0.01.

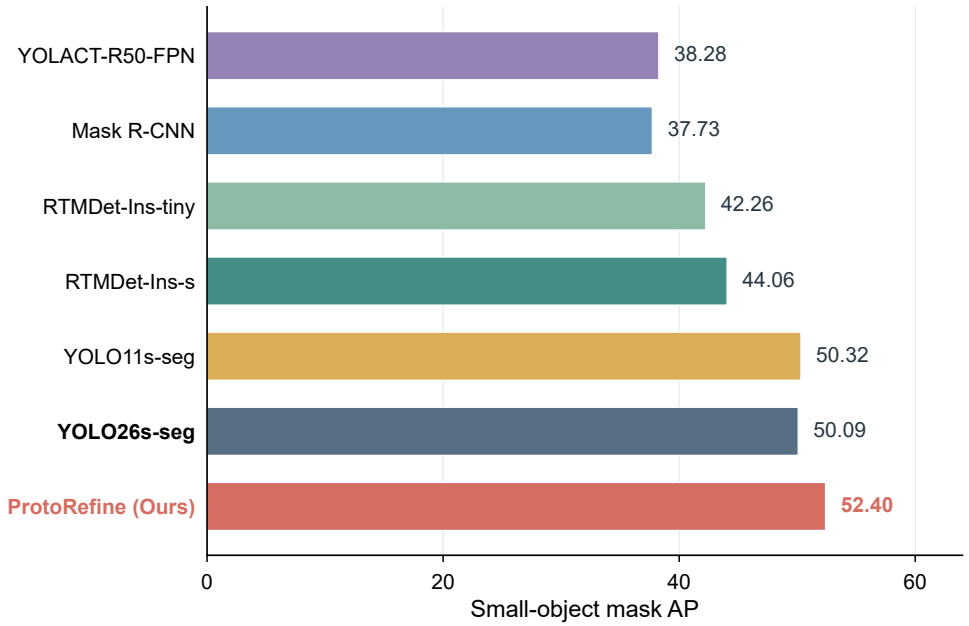


Fig. S10 Small-object mask quality across complete systems. All seven point estimates use the shared 265-image development split at 640 pixels and correspond to Table S12.

The additional examples were selected to illustrate failure and little-change outcomes, not their frequencies. Each additional row shows the same matched target and crop across methods, with ProtoRefine beside ground truth. Green marks the annotated contour and coral the predicted foreground; masks are enlarged without smoothing or manual correction. Main Figure 5 displays all candidates above score 0.05 using box-correspondence colors.

11 Published strawberry benchmark context

Table S13 places the present official-test result alongside published StrawDI reports. AP/mAP values are transcribed from the original studies, retaining their training, initialization and inference settings rather than forming a common-protocol ranking. Training/test counts denote images. For Centroid FCN (†), only the publisher-abstract score could be verified; full split and evaluator details were unavailable. The published Mask R-CNN/TensorRT study is separate from our Mask R-CNN development rerun. Direct comparisons within this study are reported by evaluation split in main Table 1.

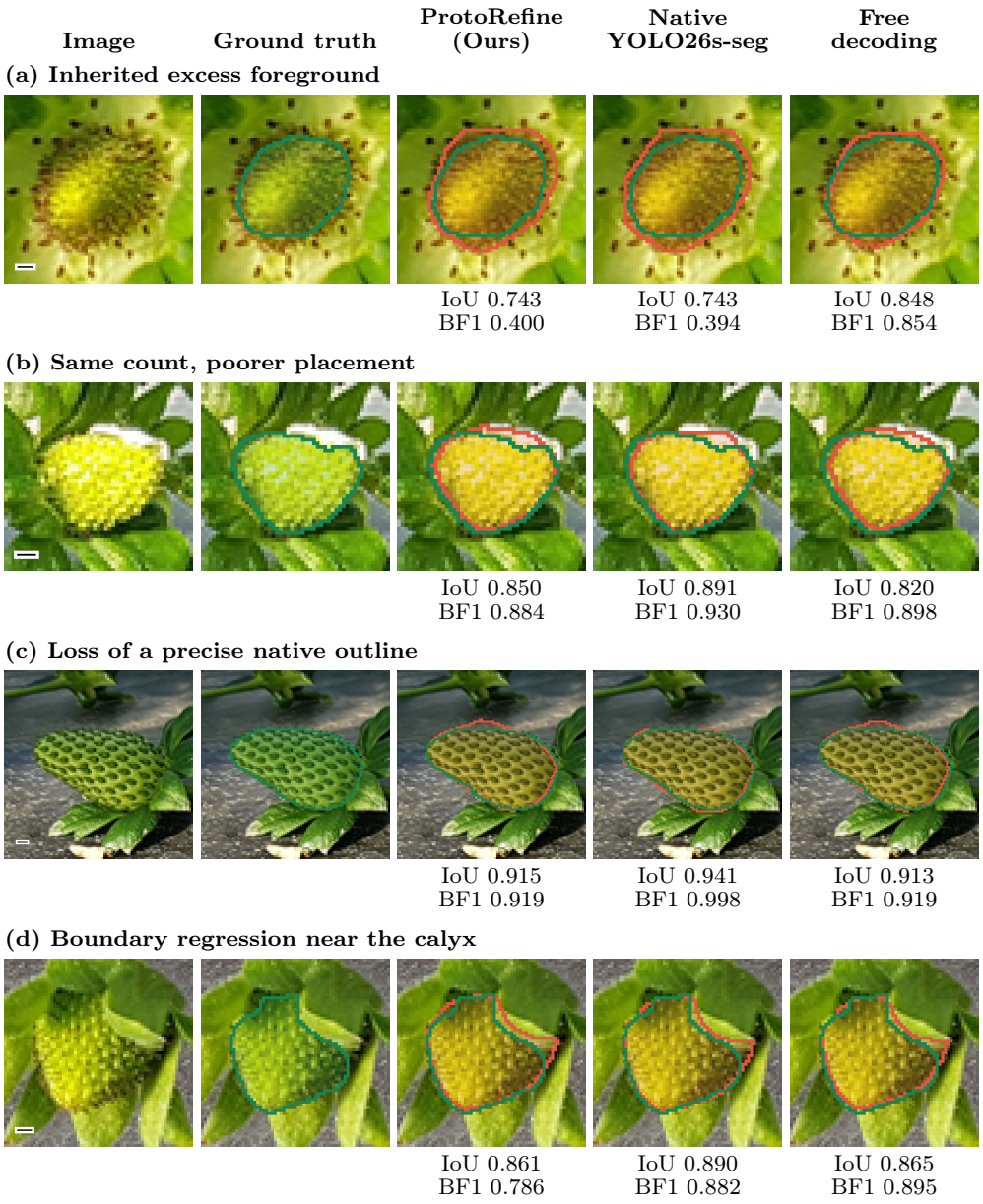


Fig. S11 Selected refinement failures. (a) The inherited native count retains excess foreground that free decoding can remove. (b) Native and refined masks both have the correct foreground count, yet refinement reduces overlap. (c,d) Refinement also worsens an initially accurate outline, including a calyx-adjacent boundary. Green denotes the annotated contour and coral the predicted mask. Scale bars are five original-image pixels; BF1 is boundary F1. These selected cases illustrate limitations, not their frequency.

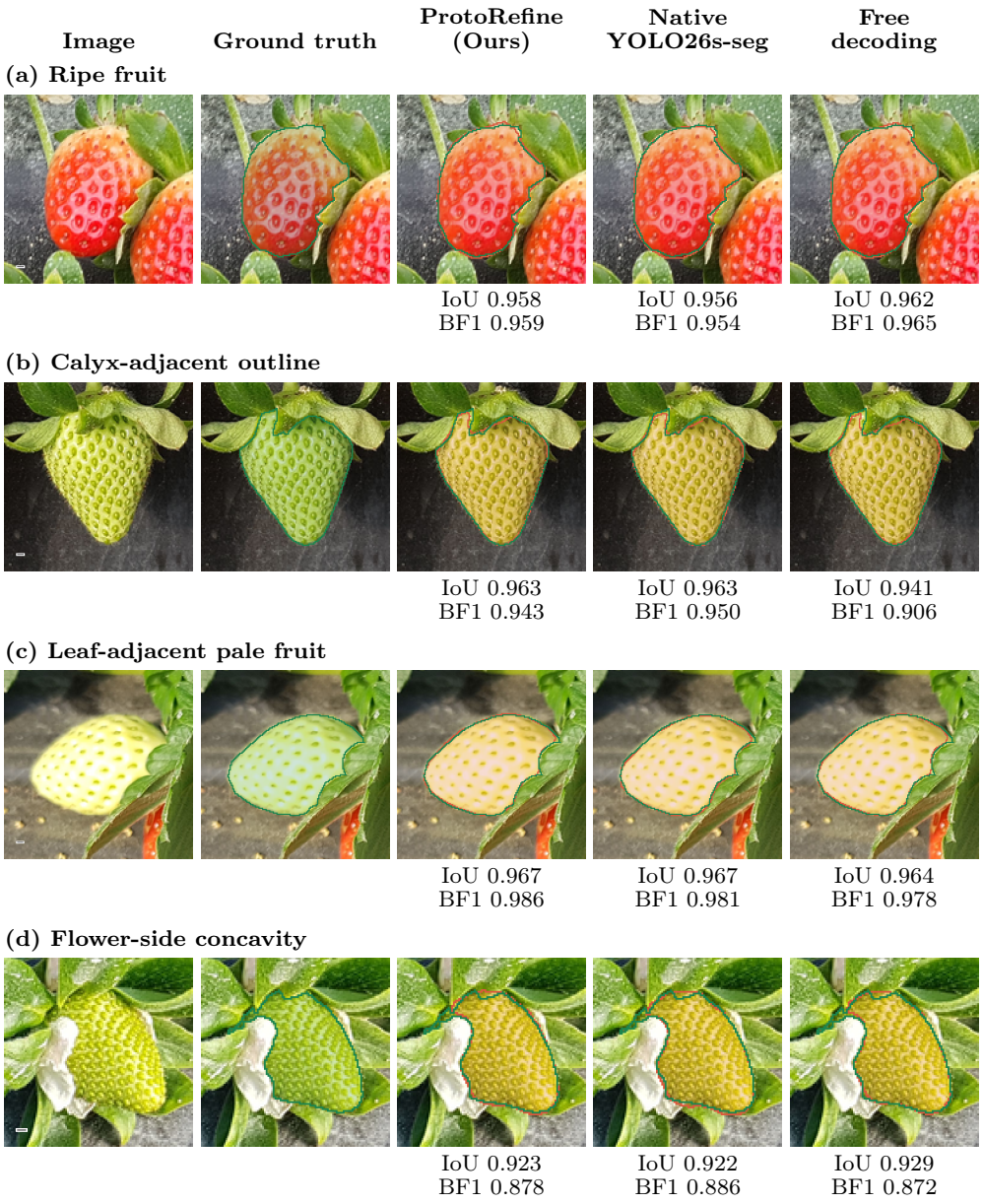


Fig. S12 Selected little-change outcomes. The native masks already follow the visible fruit closely across (a) a ripe fruit, (b) a calyx-adjacent outline, (c) a leaf-adjacent pale fruit and (d) a flower-side concavity. ProtoRefine changes IoU by less than 0.2 pp in each example. Colors and five-original-pixel scale bars follow Figure S11. The four source images differ from those in the failure figure and main-paper examples.

Method	Year	Main design	AP/mAP	Training/ test
Published studies				
Pérez-Borrero et al. (2020)	2020	Detection-guided masks	43.85	2800/200
Centroid FCN (Perez-Borrero et al. 2021)	2021	Centroid grouping	52.61[†]	–[†]
StrawSeg (Cao et al. 2023)	2023	MobileNetV2 / FAN	80.0	2800/200
Mask R-CNN R50-FPN (Crespo et al. 2025)	2025	Region-based masks	83.46	2800/200
Mask R-CNN / TensorRT (Crespo et al. 2025)	2025	INT8 deployment	83.17	2800/200
Current study				
ProtoRefine	2026	Prototype-conditioned refinement	84.03	2427/200

Table S13 Published StrawDI results under their original study settings.

References

Bolya D, Zhou C, Xiao F, et al (2019) YOLACT: Real-time Instance Segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, URL https://openaccess.thecvf.com/content_ICCV_2019/papers/Bolya_YOLACT_Real-Time_Instance_Segmentation_ICCV_2019_paper.pdf

Cao L, Chen Y, Jin Q (2023) Lightweight Strawberry Instance Segmentation on Low-Power Devices for Picking Robots. Electronics 12(14):3145. https://doi.org/10.3390/electronics12143145, URL https://www.mdpi.com/2079-9292/12/14/3145

Crespo A, Moncada C, Crespo F, et al (2025) An efficient strawberry segmentation model based on Mask R-CNN and TensorRT. Artificial Intelligence in Agriculture 15(2):327–337. https://doi.org/10.1016/j.aiia.2025.01.008, URL https://www.sciencedirect.com/science/article/pii/S258972172500008X

He K, Gkioxari G, Dollár P, et al (2017) Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, URL https://arxiv.org/abs/1703.06870

Jocher G, Qiu J, Liu M, et al (2026) Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. arXiv:2606.03748v1, URL https://arxiv.org/abs/2606.03748

Lyu C, Zhang W, Huang H, et al (2022) RTMDet: An Empirical Study of Designing Real-Time Object Detectors. arXiv:2212.07784, URL https://arxiv.org/abs/2212.07784

Pérez-Borrero I, Marín-Santos D, Gegúndez-Arias ME, et al (2020) A fast and accurate deep learning method for strawberry instance segmentation. Computers and Electronics in Agriculture 178:105736. https://doi.org/10.1016/j.compag.2020.105736, URL https://zenodo.org/records/14658929/files/PerezBorrero_et_al_2020_PostPrint.pdf?download=1, author accepted manuscript

Perez-Borrero I, Marin-Santos D, Vasallo-Vazquez MJ, et al (2021) A new deep-learning strawberry instance segmentation methodology based on a fully convolutional neural network. Neural Computing and Applications 33:15059–15071. https://doi.org/10.1007/s00521-021-06131-2, URL https://link.springer.com/article/10.1007/s00521-021-06131-2