Online Data Curation for Object Detection via Marginal Contributions to Dataset-level Average Precision

Supplementary Material

Appendix 1. Ablation studies on teacher with asynchronized architecture. We tested FasterRcnn-Res50 with teachers with asynchronous backbones or model families. We found that the performance drops if the teacher does not remain within the same model family. As shown in Table 1, asynchronous curation across very different architectures often yields weaker gains. This observation echoes the knowledge-distillation literature, where architectural mismatch leads to a knowledge gap between teacher and student. Hence, we generally select the same architecture but a larger backbone (Res101/Res152) as the default teacher for data curation.

Table 1. **Asynchronous teacher-student pairing.** Different teacher architectures yield weaker gains than same-family teachers (student: Faster R-CNN–Res50, baseline 37.4).

Teacher (for scoring)	Teacher mAP	Student mAP
Faster RCNN–ResX101 [7]	42.5	38.9
Faster RCNN–SwinL [4]	49.4	38.6
Faster RCNN–Res101 [5]	41.8	**39.9**
Deform. DETR–Res50 [10]	46.2	38.1
DINO–Res50 [9]	49.0	38.3

Appendix 2. Ablation studies on low quality data. Section 4.3 in the main text analyzes DetGain under artificially corrupted training data. To evaluate robustness under low-quality data, we construct noisy variants of the COCO training set following a simple corruption pipeline. For each image, noise is injected with probability $p \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$. We apply four corruption types:

- Bounding box jittering: randomly scale each box $[x, y, w, h]$ by factors $s_w, s_h \in [1{-}0.5,\ 1{+}0.5]$, enforcing a minimum deviation of 5%. The box is re-centered and clipped to the image.
- Label noise: uniformly replace the category of a random subset of boxes, affecting a fraction drawn from $[0.2,\ 0.5]$ of instances in the image.
- Deletion: randomly remove a fraction $[0.2,\ 0.5]$ of ground-truth boxes.
- Fake box addition: insert m synthetic boxes with $m \sim \lfloor n \cdot r \rfloor$, $r \in [0.2,\ 0.5]$, size sampled as $w \in [0.05W,\ 0.2W]$, $h \in [0.05H,\ 0.2H]$, and reject if $\max_a \mathrm{IoU}(\mathrm{box}, a) \geq 0.1$. We cap at 20 fake boxes per image.

Each p yields an independent annotation file generated in an offline manner.

We further evaluate the DetGain on pseudo-labeled datasets by incorporating 120k additional COCO-unlabeled images [3]. Specifically, we use YOLOv8-nano [6] to generate pseudo-labels for the unlabeled images and adopt a Faster R-CNN-Res152 teacher to perform data curation while training a Faster R-CNN-Res50 student. As summarized in Table 2, DetGain again yields consistent improvements under these automatically generated, noisy labels. When the pseudo-labeled data are combined with the clean annotations, performance further increases—resembling a semi-supervised learning regime. However, we simply include the pseudo-labeled samples in the curated training set without employing any specialized semi-supervised learning techniques. This result indicates that DetGain can flexibly cope with both random annotation noise and model-induced label noise. Notably, we intentionally avoid using the same architecture for pseudo-label generation and for student training, since identical models tend to share systematic errors that are difficult to disentangle during data scoring.

Table 2. **Pseudo-labeling results with YOLOv8 [6].** Teacher for data curation: Faster R-CNN–Res152. Student: Faster R-CNN–Res50. DetGain improves both clean and pseudo-labeled data, showing robustness to model-generated label noise.

Dataset	Baseline	+Aug	+DetGain
Train	37.4	37.5	**40.0** (+2.6)
Unlabeled	32.8	32.6	**35.6** (+2.8)
Unlabeled + Train	37.6	36.6	**40.6** (+3.0)

Appendix 3. Additional mathematical derivations

A3.1 Preliminaries and notation. For simplicity, we define all notations in Table 3. The (non-interpolated) AP has the Stieltjes form

$$\mathrm{AP} = \int_0^1 p(u)\, dr(u). \tag{1}$$

Since $r(u) = \frac{T}{T_{\mathrm{GT}}}\big[1 - F_{\mathrm{TP}}(u)\big]$, we have $dr(u) = -(T/T_{\mathrm{GT}})\, f_{\mathrm{TP}}(u)\, du$, yielding

$$\mathrm{AP} = -\frac{T}{T_{\mathrm{GT}}} \int_0^1 p(u)\, f_{\mathrm{TP}}(u)\, du. \tag{2}$$

A3.2 Single-insertion ΔAP: general derivation. Insert a *single detection* with score s into the ranked list. De-

Symbol	Definition / Domain
$u \in (0,1)$	Score threshold scanned from high to low.
$\mathcal{D}$	Current dataset / pool used to accumulate counts.
$T_{\mathrm{GT}} \in \mathbb{N}$	Total number of ground-truth instances over $\mathcal{D}$.
$T, F \in \mathbb{N}$	Dataset-level counts of true/false positives already accumulated from $\mathcal{D}$.
$A = T + F$	Total number of positives (TP+FP) accumulated so far.
$F_{\mathrm{TP}}(u)$, $f_{\mathrm{TP}}(u)$	CDF/PDF of TP scores (w.r.t. u).
$F_{\mathrm{FP}}(u)$, $f_{\mathrm{FP}}(u)$	CDF/PDF of FP scores (w.r.t. u).
$C_{\mathrm{TP}}(u)$	Cumulative TPs *above* threshold u: $C_{\mathrm{TP}}(u) = T\,[1 - F_{\mathrm{TP}}(u)]$.
$C_{\mathrm{FP}}(u)$	Cumulative FPs *above* threshold u: $C_{\mathrm{FP}}(u) = F\,[1 - F_{\mathrm{FP}}(u)]$.
$N(u)$	Total predictions above u: $N(u) = C_{\mathrm{TP}}(u) + C_{\mathrm{FP}}(u)$.
$p(u)$	Precision at u: $p(u) = \dfrac{C_{\mathrm{TP}}(u)}{N(u)}$.
$r(u)$	Recall at u: $r(u) = \dfrac{C_{\mathrm{TP}}(u)}{T_{\mathrm{GT}}}$.
$\mathcal{C}, \mathcal{T}$	Class set and IoU-threshold set for mAP aggregation.

Table 3. Notation used in Appendix A3 (dataset-level scoring and AP integration).

note the updated precision/recall by $p'(u), r'(u)$ and $\mathrm{AP}' = \int_0^1 p'(u)\,dr'(u)$. Decompose

$$\delta\mathrm{AP} = \mathrm{AP}' - \mathrm{AP} = \underbrace{\int_0^1 p'(u)\,d\big[r'(u) - r(u)\big]}_{\text{(I)}} + \underbrace{\int_0^1 \big[p'(u) - p(u)\big]\,dr(u)}_{\text{(II)}}. \tag{3}$$

Term (I) accounts for the *impulse* in recall if a TP is inserted at $u = s$; term (II) accounts for *precision reweighting* on the existing TP measure $dr(u)$.

Case (i): insert a TP with score s. For $u \le s$, $C'_{\mathrm{TP}} = C_{\mathrm{TP}} + 1$ and $N' = N + 1$; for $u > s$ nothing changes. Recall gains a point mass $1/T_{\mathrm{GT}}$ at $u = s$:

$$d\big[r'(u) - r(u)\big] = \frac{1}{T_{\mathrm{GT}}}\,\delta(u - s)\,du. \tag{4}$$

Hence the *self-contribution* is

$$\text{(I)} = \frac{1}{T_{\mathrm{GT}}}\,p'(s) = \frac{1}{T_{\mathrm{GT}}}\,\frac{C_{\mathrm{TP}}(s) + 1}{N(s) + 1}. \tag{5}$$

For term (II), only $u \le s$ matters. Using

$$\begin{aligned} p'(u) - p(u) &= \frac{C_{\mathrm{TP}}(u) + 1}{N(u) + 1} - \frac{C_{\mathrm{TP}}(u)}{N(u)} \\ &= \frac{N(u) - C_{\mathrm{TP}}(u)}{N(u)\,[N(u) + 1]} = \frac{C_{\mathrm{FP}}(u)}{N(u)\,[N(u) + 1]}, \end{aligned} \tag{6}$$

and $dr(u) = -(T/T_{\mathrm{GT}})\,f_{\mathrm{TP}}(u)\,du$, we obtain

$$\text{(II)} = \frac{T}{T_{\mathrm{GT}}} \int_0^s \frac{C_{\mathrm{FP}}(u)}{N(u)\,[N(u) + 1]}\,f_{\mathrm{TP}}(u)\,du. \tag{7}$$

Combining (5)–(7):

$$\delta^{\mathrm{TP}}_{\mathrm{AP}}(s) = \frac{1}{T_{\mathrm{GT}}}\,\frac{C_{\mathrm{TP}}(s) + 1}{N(s) + 1} + \frac{T}{T_{\mathrm{GT}}} \int_0^s \frac{C_{\mathrm{FP}}(u)}{N(u)\,[N(u) + 1]}\,f_{\mathrm{TP}}(u)\,du. \tag{8}$$

Case (ii): insert an FP with score s. Recall does not change, so (I)$= 0$. For $u \le s$, $C'_{\mathrm{TP}} = C_{\mathrm{TP}}$ and $N' = N + 1$,

$$p'(u) - p(u) = \frac{C_{\mathrm{TP}}(u)}{N(u) + 1} - \frac{C_{\mathrm{TP}}(u)}{N(u)} = -\frac{C_{\mathrm{TP}}(u)}{N(u)\,[N(u) + 1]}. \tag{9}$$

Thus

$$\delta^{\mathrm{FP}}_{\mathrm{AP}}(s) = -\frac{T}{T_{\mathrm{GT}}} \int_0^s \frac{C_{\mathrm{TP}}(u)}{N(u)\,[N(u) + 1]}\,f_{\mathrm{TP}}(u)\,du. \tag{10}$$

A3.3 Closed forms under a uniform prior (Beta(1,1)). For general applicability across detectors and efficient scoring, we adopt the following uniform-prior simplification:

$$f^{(c,\tau)}_{\mathrm{TP}}(u) \equiv f^{(c,\tau)}_{\mathrm{FP}}(u) \equiv 1, \qquad T_{c,\tau} = T_c^{\mathrm{GT}}, \tag{11}$$

For a fixed (c, τ), write $T = T_{c,\tau}$, $F = F_{c,\tau}$, $A = T + F$. Then:

$$\begin{cases} C_{\mathrm{TP}}(u) = T(1 - u), \\ C_{\mathrm{FP}}(u) = F(1 - u), \\ N(u) = A(1 - u). \end{cases} \tag{12}$$

(1) TP insertion: closed form for (8). Split the two terms:

$$\delta^{\mathrm{TP}}_{\mathrm{AP}}(s) = \underbrace{\frac{1}{T_{\mathrm{GT}}}\,\frac{C_{\mathrm{TP}}(s) + 1}{N(s) + 1}}_{\text{self term}} + \underbrace{\frac{T}{T_{\mathrm{GT}}} \int_0^s \frac{C_{\mathrm{FP}}(u)}{N(u)\,[N(u) + 1]}\,du}_{\text{precision reweighting}}.$$

Self term. Substitute $C_{\mathrm{TP}}(s) = T(1 - s)$ and $N(s) = A(1 - s)$:

$$\frac{1}{T_{\mathrm{GT}}}\,\frac{T(1 - s) + 1}{A(1 - s) + 1}. \tag{13}$$

Precision reweighting term. Algebraic simplification gives

$$\frac{C_{\mathrm{FP}}(u)}{N(u)\,[N(u) + 1]} = \frac{F}{A} \cdot \frac{1}{A(1 - u) + 1}.$$

Hence

$$\int_0^s \frac{C_{\mathrm{FP}}(u)}{N(u)\,[N(u)+1]}\,du = \frac{F}{A}\int_0^s \frac{du}{A(1-u)+1}.$$

Let $t = 1-u$ ($dt = -du$). As $u : 0 \to s$, $t : 1 \to 1-s$:

$$\int_0^s \frac{du}{A(1-u)+1} = \int_{1-s}^1 \frac{dt}{At+1} = \frac{1}{A}\,\ln\frac{A+1}{A(1-s)+1}.$$

Therefore

$$\frac{T}{T_{\mathrm{GT}}}\int_0^s \frac{C_{\mathrm{FP}}(u)}{N(u)\,[N(u)+1]}\,du = \frac{TF}{T_{\mathrm{GT}}A^2}\,\ln\frac{A+1}{A(1-s)+1}. \tag{14}$$

Combine (13)–(14):

$$\boxed{\delta_{\mathrm{AP}}^{\mathrm{TP}}(s) = \frac{1}{T_{\mathrm{GT}}}\left[\frac{T(1-s)+1}{A(1-s)+1} + \frac{TF}{A^2}\,\ln\frac{A+1}{A(1-s)+1}\right]} \tag{15}$$

(2) FP insertion: closed form for (10). Similarly,

$$\frac{C_{\mathrm{TP}}(u)}{N(u)\,[N(u)+1]} = \frac{T}{A}\cdot\frac{1}{A(1-u)+1}.$$

Thus

$$\int_0^s \frac{C_{\mathrm{TP}}(u)}{N(u)\,[N(u)+1]}\,du = \frac{T}{A^2}\,\ln\frac{A+1}{A(1-s)+1}.$$

Plug into (10):

$$\boxed{\delta_{\mathrm{AP}}^{\mathrm{FP}}(s) = -\frac{T^2}{T_{\mathrm{GT}}A^2}\,\ln\frac{A+1}{A(1-s)+1}} \tag{16}$$

Both (15)–(16) are $O(1)$ to evaluate and remain *monotonic* in the detection score s—increasing for true positives (TPs) and decreasing for false positives (FPs). We adopt a *uniform prior* over $(0,1)$ as a maximum-entropy, model-agnostic baseline, which yields closed-form weights without per-iteration density fitting. Importantly, our method is used for *ranking*: samples are selected by the teacher–student DetGain gap $s_{\mathrm{DG}}(x)$, so mild bias in absolute DetGain values largely cancels when taking the difference, while the desired qualitative dependencies are preserved (higher-scoring TPs contribute more positively; higher-scoring FPs contribute more negatively; and rarer classes with smaller T_c^{GT} produce proportionally larger per-class shifts).

Further simplification is possible by fixing a global TP:FP ratio (e.g., $T{:}F = 1{:}9$ on COCO) rather than re-estimating (T, F) per detector. This is motivated by the common practice that modern detectors output a fixed number of predictions per image (e.g., 100), while COCO images contain roughly $\sim$10% ground-truth instances per prediction budget on average. Crucially, any fixed $T{:}F$ ratio only rescales the same monotonic functions in (15)–(16) and therefore does not affect the *relative ordering* that drives selection.

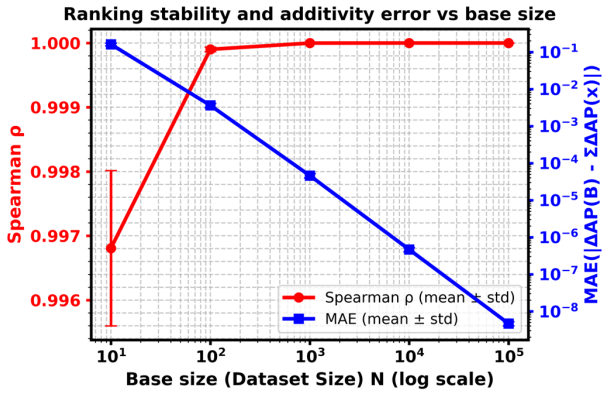

Figure 1. **Ranking stability and additivity error vs. base set size.** We simulate the super-batch selection setting (e.g., $64 \to 16$) using fitted detector outputs at a real checkpoint. For each candidate batch B, we compare the additive estimate $\hat{s}(B) = \sum_{x\in B}\Delta AP(x)$ with the true batch gain $s(B) = AP(\mathcal{D}\cup B) - AP(\mathcal{D})$. As the base dataset size $|\mathcal{D}|$ increases, ranking agreement quickly approaches 1 (left axis: Spearman ρ, mean±std), while the additivity error decreases (right axis: $\mathrm{MAE}(|\Delta AP(B) - \sum_x \Delta AP(x)|)$, mean±std).

A3.4 Assumptions and first-order additivity. Our batch score uses a first-order expansion of the evaluation metric around the current base set $\mathcal{D}$: for a small selected subset $\mathcal{B}$ from a large super-batch,

$$\delta_{\mathrm{mAP}}(\mathcal{B}; f, \mathcal{D}) = \sum_{x\in\mathcal{B}} \delta_{\mathrm{mAP}}(x; f, \mathcal{D}) + O\left(\frac{|\mathcal{B}|}{|\mathcal{D}|}\right). \tag{17}$$

In our setting, $\mathcal{B}$ is formed by ranking candidates inside a small sub-batch (e.g., selecting k=20% from a super-batch), while $\mathcal{D}$ is the full training set (typically orders of magnitude larger), thus $|\mathcal{B}| \ll |\mathcal{D}|$ and the remainder term is negligible *for ranking*.

To quantify the magnitude of ignored interactions, we simulate the exact super-batch selection protocol used in training (e.g., $64 \to 16$) using fitted detector outputs at a real checkpoint. For each candidate batch B, we compare the additive estimate $\hat{s}(B) = \sum_{x\in B}\Delta AP(x)$ with the true batch gain $s(B) = AP(\mathcal{D}\cup B) - AP(\mathcal{D})$. As $|\mathcal{D}|$ increases, the interaction becomes rapidly negligible: the ranking agreement between $\hat{s}(B)$ and $s(B)$ quickly approaches 1 (Spearman $\rho \approx 1$; see Fig. 1). This supports the validity of using Eq. (17) as a *ranking surrogate* within each super-batch. DetGain is computed from *post-processed* predictions (after each detector's decoding and NMS), so intra-image non-linear interactions induced by NMS are already reflected in the final TP/FP set and their confidence scores. The sanity check in Fig. 1 also indicates that any residual cross-sample interaction does not destabilize the batch ranking in the regime where $|\mathcal{B}| \ll |\mathcal{D}|$.

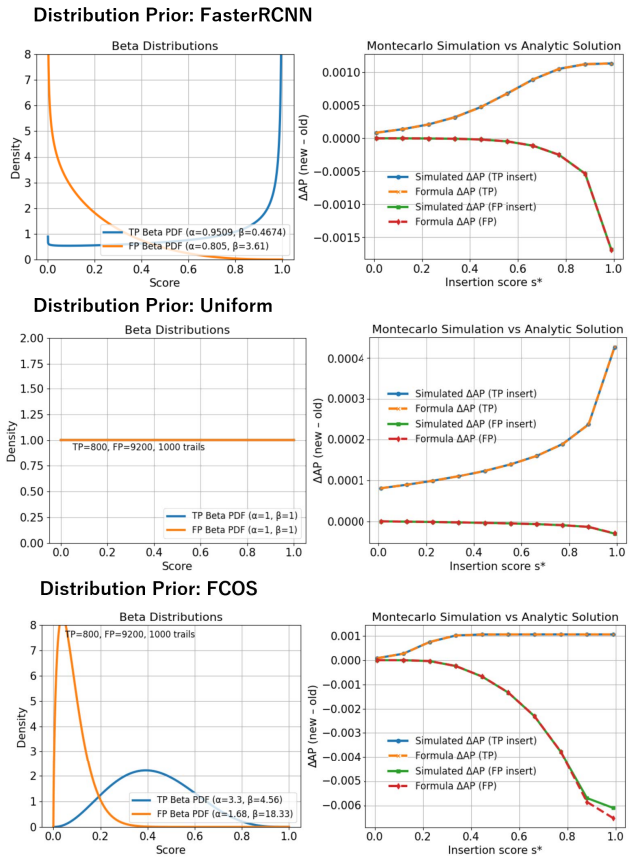

Figure 2. **Monte Carlo verification of the analytic ΔAP formulation.** Each subplot compares simulated and analytic ΔAP for TP and FP insertions under different Beta priors (top: Faster R-CNN, bottom: FCOS). The excellent agreement demonstrates the correctness and numerical stability of our closed-form derivation.

Appendix 4. Verification via Monte Carlo simulation

To verify the correctness of our analytic formulation in Eq. (8, 10), we perform a Monte Carlo simulation that directly emulates the insertion of a single detection under controlled TP/FP score distributions. For each setting of detector priors (e.g., Faster R-CNN or FCOS), we draw TP and FP scores from their respective Beta distributions, repeatedly compute dataset-level AP before and after inserting one detection with a fixed score s^*, and record the resulting ΔAP. By averaging over thousands of random trials, the stochastic estimate converges to the expected ground-truth value at the statistical level. Specifically, we first estimate the Beta distributions of TP and FP scores from each detector trained on the COCO training set and use these distributions as priors for simulation. In each run, we draw $T = 800$ TP scores and $F = 9200$ FP scores ($A = T + F$) to mimic a similar scenario to COCO. The total number of ground-truth instances is fixed to $T_{\mathrm{GT}} = 1000$. We evaluate insertion scores $s^* \in [0.01, 0.99]$ at 10 evenly spaced values. For each s^*, we repeat the Monte Carlo procedure for 1000 independent trials and record the mean $\mathbb{E}[\Delta\mathrm{AP}]$ for both cases: (i) *TP insertion* (adding a correct detection of score s^*), and (ii) *FP insertion* (adding a spurious detection of score s^*).

Fig. 2 compares the Monte Carlo results with our analytic solution obtained from Eq. (8–10). The two curves almost perfectly overlap across all score ranges and detectors, confirming that the proposed closed forms faithfully capture the expected marginal change in AP. While Monte Carlo simulation requires extensive sampling—thousands of random draws per score and hundreds of repetitions—our analytic expression only needs a lightweight one-dimensional numerical integration. In practice, using roughly 300 uniformly sampled points is sufficient to obtain a numerically stable ΔAP estimate within 10^{-4} of the Monte Carlo result, offering over three orders of magnitude faster computation.

Appendix 5. Implementation details. We implement *DetGain* as a thin, model-agnostic wrapper around the training loop. The curator does not modify the detector per se. It only (i) queries a standard `predict(...)` interface on a *super-batch*, (ii) computes per-image metric-driven *DetGain* for both student and teacher, and (iii) slices/reorders the super-batch into a *sub-batch* for the actual gradient step. Because DetGain consumes only common detection outputs (boxes, scores, labels, and matched IoUs), it works across one-stage/two-stage/transformer detectors, across different codebases (e.g., MMDetection/Detectron2), and with heterogeneous teacher-student pairs.

Teacher choice. We require a pretrained teacher optimized on the same training set. In principle, the teacher can come from any architecture family, independent of the student. In practice, we find that using a teacher from the same family with a stronger backbone (e.g., ResNet-101/152), and trained with same schedules gives the most stable gains.

Data pipeline and augmentation. The data loader applies the baseline pipeline (resize/flip/crop/photometric, etc.) to produce a *strong-augmented* super-batch of size B. DetGain selection is performed on this super-batch to obtain indices of the top-$k = \lfloor B \cdot \rho \rfloor$ images, $\rho \in (0, 1]$. The per-iteration procedure is summarized in Algorithm 1.

To improve data diversity and generalization, we adopt a unified augmentation framework that integrates geometric, photometric, and compositional transformations. All augmentations are implemented within the MMDetection pipeline, combining `Albumentations` [1] and `Copy-Paste` modules. The same augmentation pool is shared across all detectors, while the application probabilities and strengths are roughly adjusted per model family, as we observed different detectors exhibit different generalization abilities

Table 4. **Summary of strong data augmentation components.** Each operator is independently sampled with the listed probability (%). Probabilities are adjusted by detector family: one-stage (lower), two-stage (default), and transformer-based (higher). To enlarge the augmentation space, we combine three branches: plain (37.5%), strong (37.5%), and Copy–Paste (25%).

Category	Augmentation	Branch	Typical Param.	Prob. (%)
Geometric	RandomCrop (MinIoU)	Copy–Paste, Strong	Crop by IoU threshold range $[0.1, 0.9]$	20–30
	RandomResize	All	Multi-scale resize, short side 480–960 px	100
	RandomFlip	All	Horizontal or vertical flip ($p = 0.5$)	50
	RandomAffine	Strong	Rotation $\pm 10°$, scale $[0.8, 1.2]$, shear $\pm 6°$	20
	Copy–Paste	Copy–Paste, Strong	Paste up to 7 instances per image with valid masks	25–40
Photometric	Brightness / Contrast	All	Intensity and contrast shift (± 0.3)	30–40
	Hue–Saturation–Value	All	Color tone shift (± 0.3)	20
	CLAHE	Strong	Local contrast equalization (clip limit $= 2.0$)	15
	Posterize	Strong	Reduce bit depth (2–5 bits/channel)	10–15
	Color Distortion	Strong	Standard MMDet photometric distortion	30–50
	CoarseDropout	Strong	Random black patches ($\leq 10\%$ area)	20–25
Noise / Blur / Compression	Gaussian Blur	Strong / Copy–Paste	Blur kernel size 5–15	10–20
	Motion Blur	Strong / Copy–Paste	Simulated camera motion (5–15 px)	10–20
	GaussNoise	Strong	Additive Gaussian noise (variance 10–40)	20–30
	ISO / Multiplicative Noise	Strong	Sensor/exposure noise ($\alpha \in [0.8, 1.2]$)	15–20
	Image Compression	Strong / Copy–Paste	JPEG quality $[40, 95]$	15–20

Algorithm 1 Online DetGain Curation (model-agnostic, per iteration)

Require: ratio $\rho \in (0, 1]$; student f_s; pretrained/optional teacher f_t; DetGain estimator $\mathcal{G}$; super-batch size B.

1: **Data pipeline (strong aug):** load super-batch $\{(x_i, \mathrm{GT}_i)\}_{i=1}^{B}$.
2: **No-grad & AMP:** stop gradient recording and enable mixed precision.
3: **Student prediction:** set f_s to `eval`; $\mathrm{pred}^s \leftarrow f_s.\mathrm{predict}(\{x_i\})$. ▷ No GT passed to `predict`
4: **Student DetGain:** $g^s \leftarrow \mathcal{G}(\mathrm{pred}^s, \{\mathrm{GT}_i\}) \in \mathbb{R}^B$.
5: **Teacher prediction:** set f_t to `eval`; $\mathrm{pred}^t \leftarrow f_t.\mathrm{predict}(\{x_i\})$.
6: **Teacher DetGain:** $g^t \leftarrow \mathcal{G}(\mathrm{pred}^t, \{\mathrm{GT}_i\}) \in \mathbb{R}^B$.
7: **DetGain-based learnability:** $\ell_i \leftarrow g_i^t - g_i^s,\ i = 1, \ldots, B$.
8: **Select top-k:** $k = \max(1, \lfloor \rho B \rfloor), \mathcal{I} \leftarrow \mathrm{TopK}(\ell, k)$.
9: **Re-enable grad & exit AMP.**
10: **Form sub-batch:** $\tilde{x} = \{x_i\}_{i \in \mathcal{I}},\ \tilde{\mathrm{GT}} = \{\mathrm{GT}_i\}_{i \in \mathcal{I}}$.
11: **Train step:** set f_s to `train`; $\mathcal{L} \leftarrow f_s.\mathrm{loss}(\tilde{x}, \tilde{\mathrm{GT}})$; backprop & optimize.

under data augmentation. In particular, one-stage detectors are generally less robust to strong transformations, so we apply lighter augmentations; two-stage detectors use moderate settings; and transformer-based detectors, which demonstrate higher augmentation tolerance, are trained with stronger configurations. Table 4 summarizes the main augmentation operators and their typical strength and application probabilities.

For one-stage detectors (e.g., FCOS, ATSS, SSD), geometric perturbations are disabled or downscaled to preserve localization stability. Two-stage detectors (e.g., Faster R-CNN) adopt the default probabilities listed in Table 4. Transformer-based detectors (e.g., Deformable DETR) use the full-strength configuration, introducing stronger geometric transformations and increasing the Copy–Paste and color distortion probabilities by up to 1.2×. This adaptive tuning ensures each detector family receives an appropriately strong yet stable augmentation regime while maintaining a consistent data distribution across experiments. Note that these augmentations remain relatively naïve and are not extensively optimized per model, given the vast hyperparameter space and the dynamic data selection driven by DetGain-based subsampling, which automatically emphasizes informative samples. More adaptive approaches such as online auto-augmentation or conditional data generation could serve as promising directions for future exploration.

Appendix 6. Ablation studies on prior distribution. We further compare the proposed *uniform prior* with a detector-specific prior estimated from a pretrained Faster R-CNN (ResNet-50 backbone). The Faster R-CNN–specific distribution is obtained by maximum-likelihood estimation (MLE) over all predictions from the COCO `train2017` set. Since the model's score distribution evolves during training and gradually aligns with the teacher's output, we adopt the teacher's prediction statistics as a stable approximation of the true detection prior for simulation.

Specifically, we collect all predictions across 80 classes and 10 IoU thresholds (IoU $\in [0.5, 0.95]$ with step 0.05).

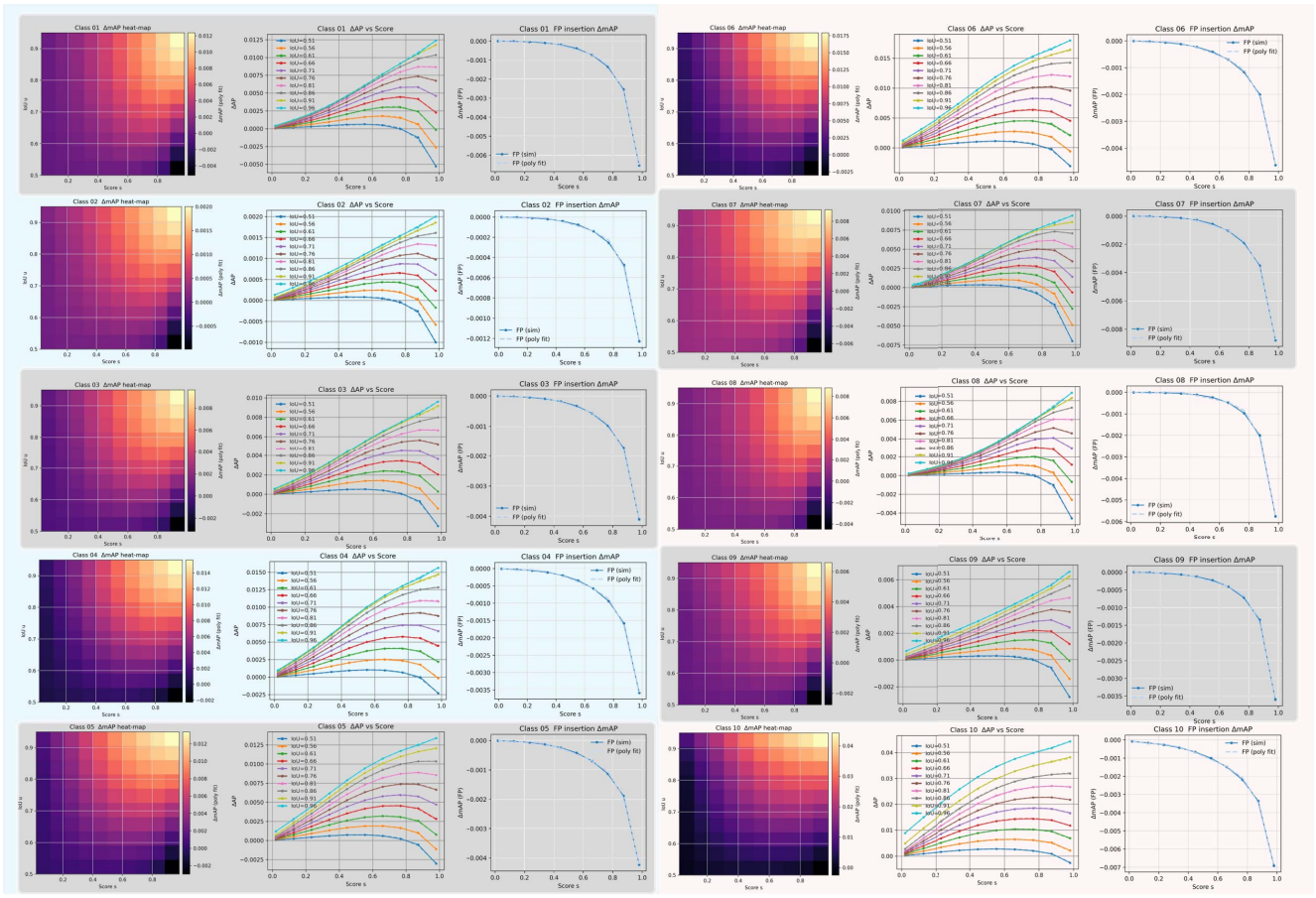

Figure 3. **Comparison between class-wise ΔAP functions under Faster R-CNN priors.** Shown are the first ten COCO classes, where each column pair visualizes the analytic ΔAP computed under the Beta-fitted prior. The 2D heatmaps (left) represent $\Delta\mathrm{AP}^{\mathrm{TP}}(s,\tau)$ with respect to score s and IoU threshold τ, while the 1D curves (right) show $\Delta\mathrm{AP}^{\mathrm{FP}}(s)$ for false-positive insertions. Both surfaces and curves are fitted with 6-order polynomials to enable efficient real-time computation of DetGain during online sampling.

Table 5. **Comparison between Faster R-CNN–specific and uniform priors.** Results are obtained using a Faster R-CNN–Res50 student and a pretrained teacher of the same architecture. The detector-specific prior is estimated from the teacher's predictions on the COCO training set.

Prior/Baseline	Train		Validation	
	AP@50	AP@[50:95]	AP@50	AP@[50:95]
Baseline	67.4	44.5	58.3	37.4
Model–specific	**68.8**	**45.6**	**61.2**	**39.6**
Uniform Prior	67.6	45.1	60.7	**39.6**

For each (c,τ) pair, we record the empirical distributions of true positives (TP) and false positives (FP) and their total counts, then fit two Beta distributions via MLE. In total, we obtain $80\times 10 = 800$ Beta parameter pairs (α,β), each describing one class–IoU condition with its corresponding $T_{c,\tau}$ and $F_{c,\tau}$. Using Eq. 8 and Eq. 10, we compute the marginal change in AP (ΔAP) under these fitted priors through numerical integration, producing per-class ΔAP surfaces with respect to IoU and score for TP insertion, and one-dimensional ΔAP curves for FP insertion. Fig. 4 visualizes the results for the first ten classes. Both can be well approximated by high-order polynomial regression to replace numerical integration for real-time computation during online sampling.

By using the above procedure, we obtain the results in Table 5. While the model–specific prior is theoretically more accurate to the teacher's score distribution and indeed yields slightly higher AP on training set, its effect on *validation* AP is negligible. We conjecture two practical reasons: (i) the fitted prior captures teacher- and training-distribution idiosyncrasies that do not transfer to held-out data; and (ii) strong online augmentation already regularizes score distributions, narrowing the gap between priors. Given the comparable convergence and final validation accuracy, together with its greater generality and zero fitting overhead,

Table 6. **Uniform-prior DetGain on additional datasets.** COCO-style mAP for Faster R-CNN-R50 on Pascal VOC2007 test and BDD100K validation using the default train/validation splits and the standard $1\times$ schedule. DetGain consistently improves over the baseline (+3.0 on VOC2007, +1.8 on BDD100K). Single run for each dataset.

Dataset	Backbone	Baseline	+DetGain	Δ
VOC2007	R50	51.3	54.3	+3.0
BDD100K	R50	30.3	32.1	+1.8

Table 7. **Runtime and memory overhead (Faster R-CNN-R50 on Pascal VOC, k=20%).** Per-iteration wall-clock time is decomposed into data loading/preprocessing and the remaining compute; the latter includes the DetGain scoring pass. Percentages are relative to the baseline.

Method	Iter time (s)	Data IO (s)	Non-data time (s)
Baseline	0.20	0.03	0.17
DetGain	0.75	0.07	0.68
Extra Overhead (%)	+275%	+133%	+300%

we adopt the *uniform prior* for all detector families and use the unified closed-form in Eqs. (15–16).

Appendix 7. Experiments on additional datasets We further evaluate the uniform-prior DetGain on datasets beyond COCO to assess robustness under different data regimes and label distributions. We report results on Pascal VOC 2007, 2012 [2] (small-scale generic detection) and BDD100K [8] (mid-scale driving with a long-tailed class distribution), using Faster R-CNN with ResNet-50 and the standard $1\times$ training schedule. We use the default training/validation splits: Pascal VOC has 16,551 training images and BDD100K has 69,863 training images and 10,000 validation images.

All other settings follow the default detection recipe of the corresponding codebase; DetGain is applied as the sample-selection criterion within each super-batch, while the detector architecture, optimizer, and schedule are unchanged.As shown in Tab. 6, DetGain consistently improves COCO-style mAP on both datasets: on VOC2007 test (train set $\sim$16.5K), DetGain improves mAP from 51.3 to 54.3 (+3.0); on BDD100K val (train set $\sim$70K), DetGain improves mAP from 30.3 to 32.1 (+1.8). These gains support that the uniform-prior DetGain remains effective across dataset scales and distribution shifts.

Appendix 8. Computational overhead and dynamic sampling. DetGain introduces an additional selection stage per iteration: we run an extra `no_grad` + AMP scoring pass on the super-batch to compute per-image DetGain scores and select the top-k images. The subsequent forward/backward update on the selected sub-batch is identical to the baseline. Therefore, peak GPU memory usage remains largely unchanged (as it is dominated by the backward pass), while the wall-clock overhead is mainly determined by the super-batch scoring cost and the selection ratio k.

We report a specific case of training Faster R-CNN-R50 on Pascal VOC with k=20%, teacher = Faster R-CNN-R50, and batch size = 16. Tab. 7 reports iteration-level timing. With a naive implementation, the total iteration time increases from 0.20s to 0.75s. Part of this increase comes from data loading and preprocessing due to reading the larger super-batch and applying strong data augmentation (0.03s $\rightarrow$ 0.07s). After subtracting this I/O difference, the remaining overhead can be attributed to the scoring-and-selection stage (teacher inference, student inference, and DetGain computation), which costs approximately 0.51s per iteration.

GPU peak memory differs by less than 5% between the baseline and DetGain. CPU memory increases from 4009MB to 6725MB (approximately +67%), mainly due to buffering the larger super-batch.

Effect of longer training schedules. DetGain's improvements are not simply an artifact of additional compute. On FCOS-R50 trained on COCO, simply extending the baseline training schedule yields diminishing returns and can even degrade validation performance due to overfitting: the baseline saturates at $2\times$ (38.5 mAP) and drops at $3\times$ (37.1 mAP), whereas DetGain continues to improve monotonically from $0.5\times$ to $3\times$ (37.5 $\rightarrow$ 43.8 mAP; Tab. 8). This suggests that DetGain alters the optimization trajectory and mitigates late-stage overfitting, rather than merely benefiting from longer training.

Efficient Implementation. To reduce overhead while preserving performance, we further explore a *dynamic* selection ratio schedule. Specifically, we use a larger ratio early in training to encourage more diverse representation learning, and a smaller ratio later for cost-efficient fine-tuning. Concretely, we set k=40% for the first 60% of iterations and k=20% for the remaining 40%. On FCOS-R50 with a $2\times$ schedule, this dynamic strategy reaches 42.5 mAP, retaining 98.8% of the static-20% DetGain performance (42.5 vs. 43.0 mAP), while reducing the total training time by approximately 30% in practice.

We also observed that several engineering strategies (e.g., reducing the NMS candidate number during inference or performing low-resolution inference for the scoring stage) can significantly improve training speed while still maintaining a meaningful improvement over the naive implementation. We leave a systematic exploration of these optimizations for future work.

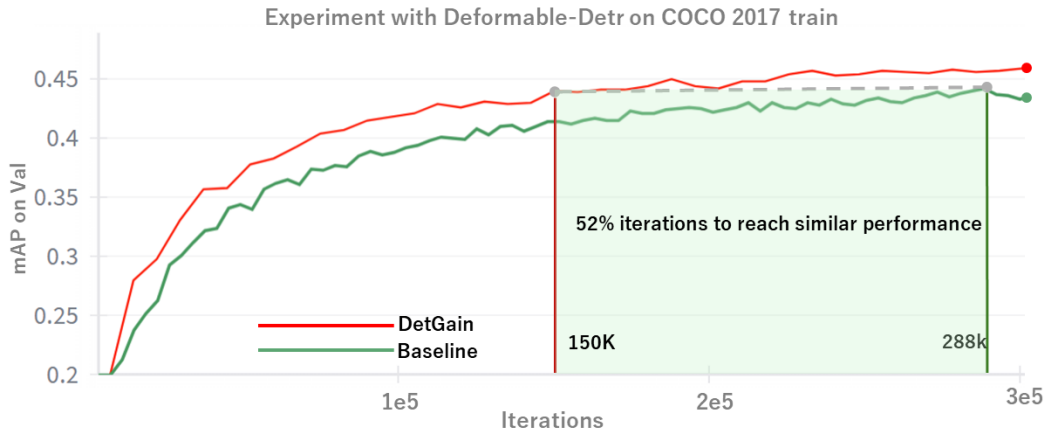

Figure 4. **Performance curve with data curation.** We compare the performance curves of the baseline model and the model with data curation. The tested model is Deformable-DETR-Res50, trained with a batch size of 16 using the standard AdamW optimizer and a fixed learning rate of 1e-4. Data curation achieved the baseline's best performance in nearly half the number of iterations.

Table 8. **Effect of longer training schedules on FCOS-R50 (COCO val mAP).** Baseline training saturates and overfits with longer schedules (performance drops at 3×), while DetGain consistently improves across schedules.

Strategy	0.5×	1×	2×	3×
Baseline	33.4	38.2	38.5	37.1 (overfit)
DetGain	37.5 (+4.1)	40.9 (+2.7)	43.0 (+4.5)	43.8 (+6.7)

Polynomial weight lookup. To simplify the online calculation, we per-fit lightweight polynomial surrogates for the DetGain weights and evaluate them in $O(1)$ per prediction. Specifically, we approximate the TP weight by a bivariate polynomial in IoU u and score s, and the FP weight by a univariate polynomial in score s:

$$w_{\mathrm{TP}}(u,s,\ell) = \sum_{p+q\le n} c_{p,q}^{(\ell)}\, u^p s^q, \; w_{\mathrm{FP}}(s,\ell) = \sum_{r=0}^{m} d_r^{(\ell)}\, s^r.$$

Coefficients $\{c_{p,q}^{(\ell)}\}$ and $\{d_r^{(\ell)}\}$ are fitted offline, following the pre-defined the analytic solution from uniform-prior.

References

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. 4

[2] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 7

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[4] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 1

[6] Rejin Varghese and M Sambath. Yolov8: A novel object detection algorithm with enhanced performance and robustness. In *2024 International conference on advances in data engineering and intelligent computing systems (ADICS)*, pages 1–6. IEEE, 2024. 1

[7] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 1

[8] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020. 7

[9] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[10] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. 1