SEEING GLOBALLY, DETECTING LOCALLY: INJECTING AND CHARACTERISING FROZEN VISION TRANSFORMER PRIORS IN OBJECT DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional neural networks (CNNs) provide efficient local inductive biases, while Vision Transformers (ViTs) aggregate global scene context via self-attention. Efforts to unify these complementary strengths into hybrid models often require complex, architecture- or domain-specific coupling, limiting transfer across detector families and obscuring how the two priors interact. To address this, we introduce a lightweight framework that uses a frozen vision foundation model as an auxiliary feature supplier, injecting its final patch representations in one direction into the deepest stage (C5) of a host convolutional backbone. The downstream pipeline remains unchanged, so the host's own detection loss determines how local and global features are mixed. Using DINOv3 as the frozen encoder, we conduct a factorial study of host initialisation and prior availability across one- and two-stage detectors, stratified data regimes on PASCAL VOC and Palm Oil, and full-scale COCO. Across settings, frozen priors consistently accelerate learning. They substitute for and complement conventional pretraining: randomly initialised hybrids surpass pretrained baselines at full PASCAL VOC on both hosts, and pretrained hosts still gain. These benefits depend on host competence and target domain. On PASCAL VOC, pretrained hybrids outperform their baselines by 7.1 AP on YOLOv8s and 13.8 AP on Faster R-CNN at full data, and the Faster R-CNN hybrid matches its full-data baseline with an estimated 11.4$\times$ fewer labels. On Palm Oil, this advantage contracts as data increase. We further identify a failure mode in which fine-tuning a COCO-pretrained YOLO detector at its default learning rate erodes accuracy on categories it already detects, with limited data restricting recovery. Overall, this study characterises the learning dynamics that emerge when frozen ViT priors meet trainable CNN representations.

1 INTRODUCTION

Convolutional neural networks (CNNs) remain the workhorse of object detection because of their efficient local inductive bias, and their hierarchical representations directly support the multi-scale features required for dense prediction (He et al., 2016; Lin et al., 2017). Their broader spatial context, however, must be assembled through successive local operations. Vision Transformers (ViTs) provide the complementary ability to model long-range interactions directly through self-attention (Dosovitskiy et al., 2021), and some of their attention heads integrate global information even in the earliest layers (Raghu et al., 2021). Recent Fourier analysis showed that self-attention exhibits a predominantly low-pass tendency (smoothing effect), whereas convolution preserves and amplifies higher-frequency structure (Park & Kim, 2022). Hybrid architectures likewise show that retaining local convolutional features alongside global transformer representations can benefit recognition and detection (Wu et al., 2021; Srinivas et al., 2021; Peng et al., 2023). Modern self-supervised foundation models (Caron et al., 2021; He et al., 2022; Oquab et al., 2024), and DINOv3 in particular (Siméoni et al., 2026), make such global representations available from large-scale pretraining.

The practical question is therefore not which representation should replace the other, but how to retain the advantages of both. A plain ViT does not natively provide the hierarchical features required

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

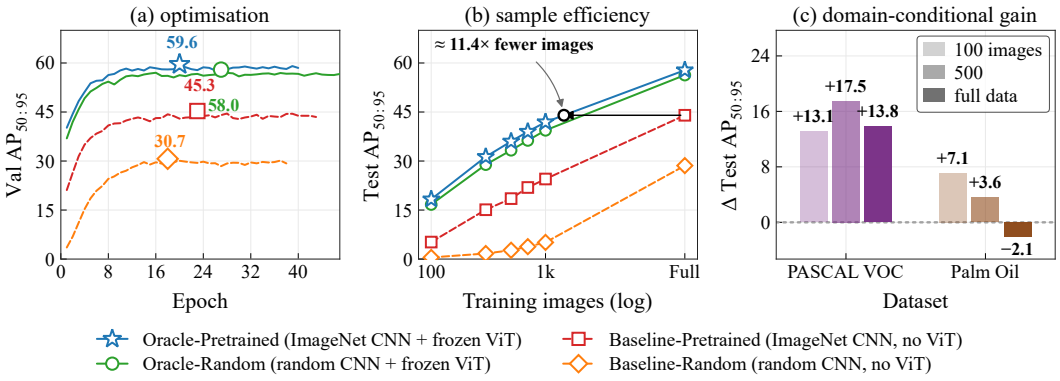

Figure 1: **Frozen ViT priors alter optimisation, sample efficiency, and their value with increasing supervision.** *Oracle* denotes an SGDL-augmented detector and *Baseline* its standalone host. The example uses Faster R-CNN (ResNet-18-FPN) supplied at C5 by a frozen DINOv3 ViT-S/16. **(a) Optimisation (validation, PASCAL VOC):** the frozen prior accelerates learning, and a randomly initialised hybrid can overtake the conventionally pretrained baseline. **(b) Sample efficiency (test, PASCAL VOC):** the pretrained hybrid reaches the full-data baseline level with substantially fewer labelled images. **(c) Domain-conditional gain (test):** the pretrained Oracle–Baseline gap persists with increasing supervision on PASCAL VOC but contracts on Palm Oil.

by conventional detectors, so using it as the primary backbone requires additional dense-prediction machinery such as pyramids or adapters (Li et al., 2022b; Chen et al., 2023; Xia et al., 2024). Fine-tuning also makes the foundation representation task-specific and can distort the very features that made it valuable (Vasconcelos et al., 2022; Kumar et al., 2022), while increasing downstream optimisation cost. We argue that an established CNN detector already supplies the complementary structure: a mature local hierarchy, native multi-scale detection machinery, and parameters that can adapt freely to the target task. Keeping the ViT separate and frozen therefore preserves both roles—a task-adaptive local detector and a fixed global semantic prior.

We introduce the SGDL (*Seeing Globally, Detecting Locally*) framework, a minimal unidirectional interface between these pathways. A frozen ViT runs in parallel with the trainable CNN detector; its final patch representation is projected, aligned with the deepest convolutional stage (C5), fused once, and returned at the host's native channel width. The detector's neck, heads, loss and training pipeline remain unchanged.

Because the same interface serves one-stage and two-stage detectors without redesign, we also use SGDL as a controlled instrument for studying how frozen ViT priors interact with trainable CNN representations across initialisation states, supervision levels, target domains, encoders and fusion designs. We interpret the resulting data-scaling patterns through the classical approximation–estimation–optimisation decomposition.

Contributions. **[C1] We present a simple cross-detector integration method.** SGDL couples a frozen ViT to a trainable convolutional detector at a single communication site while leaving the downstream detection pipeline unchanged. The same mechanism transfers without architectural redesign across the evaluated one-stage and two-stage hosts (Section 3). **[C2] We present a controlled study of ViT–CNN fusion.** A 2×2 factorial of host initialisation and frozen prior isolates the prior from host pretraining, and ablations test the design choices (Sections 4.1–4.4). **[C3] We show that frozen global priors can substitute for and complement conventional host pretraining.** Randomly initialised hybrids can approach or surpass pretrained standalone detectors, while pretrained hosts can still gain from the same frozen representation (Section 4.2; Figure 1a). **[C4] We show that the prior's value depends on supervision and target domain.** Its advantage can persist as supervision increases or contract as the standalone host learns the task unaided. We interpret these contrasting scaling patterns through an approximation–estimation framework (Figure 1c; Section 3.2). **[C5] We identify a new fine-tuning failure mode in YOLO detectors.** Pretrained detector competence can collapse rapidly during adaptation, with the learning rate as a causal factor in the collapse (Section 5).

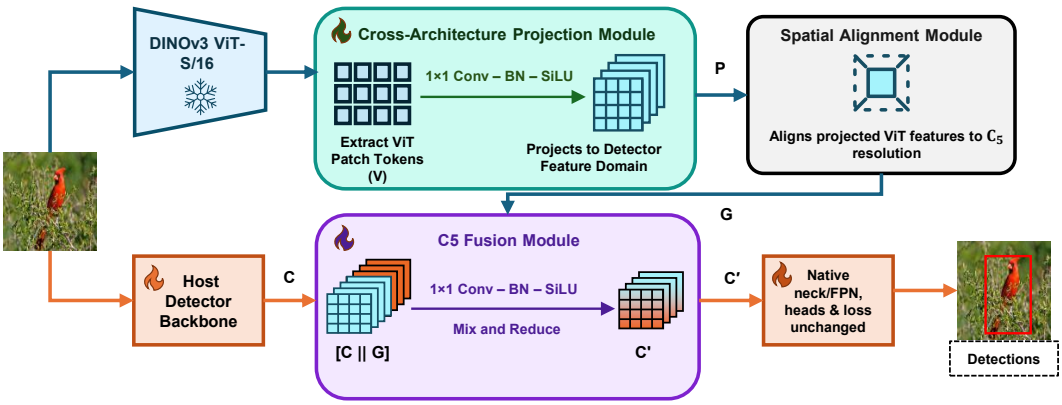

Figure 2: **SGDL fuses a frozen global prior into a trainable detector at a single site.** A frozen DINOv3 ViT-S/16 and a convolutional host process the same image in parallel. The encoder's patch-token tensor $\mathbf{V}$ is projected to $\mathbf{P}$ and pooled to the C5 grid as $\mathbf{G}$, which is fused with the host's C5 tensor $\mathbf{C}$ into $\mathbf{C}'$, the tensor that replaces $\mathbf{C}$.

2 Related Work

CNN–Transformer hybrids. Hybrid architectures combine convolutional locality with global self-attention. Some insert attention into convolutional networks (Wang et al., 2018; Bello et al., 2019; Srinivas et al., 2021), others introduce convolution into transformers (Wu et al., 2021; Xiao et al., 2021; Graham et al., 2021), and parallel designs keep separate local and global streams that exchange information (Peng et al., 2023; Chen et al., 2022). Both pathways are trained jointly and their interaction is fixed by the architecture. In the taxonomy of Haruna et al. (2025), SGDL is a parallel design with late fusion, but information flows in one direction only, from a frozen transformer into a trainable CNN, and the host detector is otherwise unchanged.

Vision Transformers as detection backbones. Transformers entered detection by replacing the convolutional backbone. ViT-FRCNN attaches a Faster R-CNN head to a plain ViT (Beal et al., 2020), Swin and PVT build hierarchical transformers for dense prediction (Liu et al., 2021; Wang et al., 2021), and ViTDet derives a simple feature pyramid from the last map of a plain ViT (Li et al., 2022b). Adapters supply the spatial and multi-scale structure that plain ViTs lack (Chen et al., 2023; Xia et al., 2024), and DEIMv2 carries this approach to real-time DETRs with DINOv3 backbones (Huang et al., 2025). In these designs the transformer is the detector's backbone and is fine-tuned with it, whereas SGDL retains the convolutional backbone and uses the transformer only as a frozen auxiliary.

Frozen foundation models in detection. A frozen pretrained backbone can rival fine-tuning when the detector head is strong enough (Vasconcelos et al., 2022; Lin et al., 2022), and F-VLM and F-ViT build open-vocabulary detectors on frozen CLIP backbones (Radford et al., 2021; Kuo et al., 2023; Wu et al., 2024). The closest work instead keeps the frozen model beside a trainable detector. Frozen-DETR passes a frozen model's class token to the decoder as an image query and fuses its patch tokens into the encoder of a query-based detector (Carion et al., 2020; Fu et al., 2024). VMCNet uses a frozen CLIP ViT to modulate the multi-scale features of a trainable CNN for open-vocabulary detection (Gao et al., 2025). DINO-YOLO inserts frozen DINOv3 modules at the input and mid-backbone of YOLOv12 and finds that the best placement depends on detector scale (Malaisree et al., 2025). DSOD injects frozen DINOv2 (Oquab et al., 2024) features at three CNN stages for source-free adaptation, with adaptive weighting and a semantic regulariser (Cai et al., 2026), and DetRefiner rescores an open-vocabulary detector's outputs with a frozen DINOv3 (Okazaki et al., 2026). Each tailors its coupling to one detector family or task, and the methods agree neither on where the frozen model should enter nor on how its features should be fused, as is common in an emerging area. SGDL tests a weaker integration, with one frozen encoder, one injection site at C5, no auxiliary objective, and the same interface for one-stage and two-stage detectors, and we pair it with a controlled study of how the prior behaves. Appendix L compares these methods in detail.

3 METHOD

3.1 OVERVIEW

We introduce the *SGDL (Seeing Globally, Detecting Locally)* framework, a minimal unidirectional coupling between a trainable convolutional detector and a frozen vision foundation model. Both process the same image in parallel, at independently chosen input resolutions, and interact once, at $\mathrm{C5}$ in feature-pyramid notation (Lin et al., 2017): the host backbone's last stride-32 stage, taken before any context-aggregation module or neck. The global prior therefore meets the host's own local features rather than a pooled summary of them.

Consider a 640×640 input, a Faster R-CNN ResNet-18-FPN host and a DINOv3 ViT-S/16 encoder (Figure 2). The encoder returns 1,600 patch tokens of dimension 384 after its class and register (Darcet et al., 2024) tokens are discarded, and the host produces a $512 \times 20 \times 20$ $\mathrm{C5}$ map $\mathbf{C}$.

Projection. The tokens are reshaped into their $384 \times 40 \times 40$ grid $\mathbf{V}$, and a 1×1 Conv–BN–SiLU block maps it to the host's width, giving the $512 \times 40 \times 40$ tensor $\mathbf{P}$. The BatchNorm layer brings the LayerNorm-normalised encoder features into the same normalisation regime as the host's features.

Alignment. A parameter-free 2×2 max-pool halves the grid to the $512 \times 20 \times 20$ tensor $\mathbf{G}$, which matches $\mathbf{C}$ (Appendix B compares learned alternatives).

Fusion. $\mathbf{G}$ and $\mathbf{C}$ are concatenated along the channel axis, and a second 1×1 Conv–BN–SiLU block reduces the 1,024 channels to 512, giving the fused map $\mathbf{C}'$.

$\mathbf{C}'$ has the shape of $\mathbf{C}$ and replaces it, so the neck, heads and loss are unchanged. The interface adds only 0.72 M trainable parameters to a ResNet-18 or YOLOv8s host, whose $\mathrm{C5}$ maps both have 512 channels, and 9.18 M to a ResNet-50 host, beside the 21.6 M-parameter frozen encoder. It uses no auxiliary loss, gating or warm-up, so the host's detection loss alone determines how the two streams are mixed; Appendix M visualises the resulting map. Appendix G gives the full formulation.

3.2 INTERPRETIVE THEORETICAL FRAMEWORK

To interpret how the gain changes with the size of the training set, we use the excess-risk decomposition of Bottou & Bousquet (2007),

$$\mathcal{E} = \underbrace{\mathcal{E}_{\mathrm{app}}}_{\text{representational limit}} + \underbrace{\mathcal{E}_{\mathrm{est}}}_{\text{finite-sample penalty}} + \underbrace{\mathcal{E}_{\mathrm{opt}}}_{\text{optimisation residual}}, \tag{1}$$

whose terms are the limit of the model family, the cost of learning from finite data, and the shortfall of optimisation. We use the decomposition only to interpret the scaling trends observed in this paper and to form hypotheses, not as a formal analysis (Appendix F). Matched Baseline–Oracle pairs share one training protocol and are trained to convergence, so we treat the optimisation error as negligible; the exception, the erosion of the pretrained YOLOv8s baseline at restricted budgets (an accidental finding), is analysed in Section 5. The remaining gap then reflects approximation or estimation error. Both can produce large gains when labels are scarce, and they differ in how the gap behaves as the training set grows.

H1: Persistent gap. If the prior relaxes an approximation limit ($\mathcal{E}_{\mathrm{app}}$) that the host cannot overcome with more data, the Oracle–Baseline gap persists at the largest budgets. Such a limit is plausible where scene context aids recognition, as on PASCAL VOC.

H2: Contracting gap. If the prior instead reduces an estimation error ($\mathcal{E}_{\mathrm{est}}$) that the host eventually overcomes unaided, the gap contracts as supervision grows. This is plausible where categories differ mainly in local colour and texture, as on Palm Oil.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We run a stratified data-regime study on PASCAL VOC and Palm Oil, a scalability test on COCO and a set of ablations.

Datasets and regimes. We use PASCAL VOC (Everingham et al., 2010), an open-source Palm Oil dataset (Suharjito et al., 2025) and COCO (Lin et al., 2014). VOC contains diverse natural scenes in which broader context can aid recognition; Palm Oil is the contrasting case, with categories distinguished mainly by local appearance against a homogeneous background. On VOC and Palm Oil we train at fixed *absolute* image budgets, drawn by multi-label stratification so that every budget keeps the source class distribution. The partitions at each budget are mutually disjoint, so averaging over them measures variation between training samples, which is large at small budgets (Wang et al., 2020), rather than seed noise on one subset. These experiments share a fixed validation split of about 600 images per dataset, whereas COCO uses its standard training and validation splits (Appendix H).

Hosts and conditions. YOLOv8s (Redmon et al., 2016; Jocher et al., 2023) is the one-stage host and Faster R-CNN (Ren et al., 2017) with a ResNet-18 (He et al., 2016) feature-pyramid backbone the two-stage host; DINOv3 ViT-S/16 is the frozen prior. To separate the effect of the prior from that of host pretraining, each host is trained in a 2×2 factorial, $\{$randomly initialised, pretrained host$\} \times \{$no ViT, frozen ViT$\}$, where the *Baseline* is the host without the ViT and the *Oracle* our hybrid. The pretrained YOLOv8s is a COCO detector, and the pretrained Faster R-CNN has an ImageNet-1K backbone (Deng et al., 2009). Matched Baseline–Oracle pairs share the partition and the training protocol. Stratified runs use 640×640 inputs, and we report test $AP_{50:95}$ on the held-out split (Appendix I).

4.2 Stratified data regimes

Table 1 reports the test AP of the four conditions and Figure 3 their validation curves.

The prior accelerates learning. The hybrid reaches the baseline's best validation AP in a fraction of the baseline's epochs, often an order of magnitude fewer, and continues to improve beyond it (Figure 3). Every entry of Table 1 is a converged endpoint except the starred cells, in which the pretrained YOLOv8s baseline loses much of its accuracy within the first epochs (shaded in Figure 3), a behaviour we found by accident and analyse in Section 5. These cells report its best checkpoint, which favours the baseline.

The prior substitutes for host pretraining. Every random-initialisation gap in Table 1 is positive, and at full PASCAL VOC the randomly initialised hybrid outperforms the pretrained baseline on both hosts (56.3 against 44.0 AP on Faster R-CNN), even on YOLOv8s, whose COCO pretraining already covers every VOC category. Once the prior is present, host pretraining adds little at full data, in line with the finding that pretraining mainly accelerates convergence when target data are plentiful (He et al., 2019). With few labels on PASCAL VOC, the two remain complementary, most clearly on YOLOv8s (about 11 AP at $n = 100$, against under 2 AP on Faster R-CNN), likely because it is pretrained as a complete COCO detector, whereas Faster R-CNN receives only an ImageNet backbone. Intuitively, the frozen encoder takes over part of the role of host pretraining through a single interface, although it stays attached at inference and adds compute (Section 4.4).

The gain persists on PASCAL VOC and contracts on Palm Oil. As supervision grows, the pretrained gap behaves differently on the two datasets. On PASCAL VOC it remains large at full data on both hosts. The prior also reduces the labels needed, as the pretrained Faster R-CNN hybrid matches its baseline's full-data accuracy with $11.4\times$ fewer labelled images, interpolated log-linearly in n (Figure 1b). On Palm Oil the gap is largest with the fewest labels and generally narrows as labels increase, until Faster R-CNN trails its baseline slightly at full data; because Palm Oil partitions vary more, we read this as a trend across budgets (Appendix J). The persistent gap on VOC matches the signature of a relaxed approximation limit (H1), and the contracting gap on Palm Oil that of a reduced estimation error (H2).

Which errors the prior corrects. Fu et al. (2024) show that frozen foundation-model features in a DETR-style detector mainly improve classification rather than localisation. We test whether SGDL behaves likewise with TIDE (Bolya et al., 2020) (Table 2; Appendix E), which attributes the AP_{50} a detector loses to each error type by correcting that type alone, so lower is better. On the full-data pretrained pairs of PASCAL VOC, SGDL follows the same pattern. Classification error falls by 54% on YOLOv8s and 78% on Faster R-CNN, against 28% and 24% for localisation error. False-positive

Table 1: **Stratified data-regime comparison.** Test $AP_{50:95}$ against training-set size. B is baseline and O oracle; $_{rnd}$ and $_{pre}$ are random and pretrained host initialisation. Δ rows are oracle − baseline in AP points. Bold marks the best entry per column within a host block. AP_{50} values and standard deviations are in Appendix J. * the baseline's accuracy erodes after an early peak, so these cells report its best checkpoint (Section 5).

	PASCAL VOC						Palm Oil						
Init.	**100**	**300**	**500**	**700**	**1,000**	**Full**	**50**	**100**	**300**	**500**	**700**	**1,000**	**Full**
Host: YOLOv8s													
B_{rnd}	1.2	6.3	10.6	14.5	19.4	62.8	10.7	18.4	30.4	36.8	39.1	41.6	49.6
O_{rnd}	21.9	35.9	41.4	44.7	49.1	**73.3**	**28.0**	28.5	37.5	**41.3**	**43.4**	45.2	**49.9**
Δ_{rnd}	+20.7	+29.6	+30.8	+30.2	+29.8	+10.5	+17.3	+10.1	+7.1	+4.5	+4.3	+3.6	+0.3
B_{pre}	**33.7***	44.0*	**48.9***	50.9*	52.5*	66.1	17.7	24.7	31.8	38.0	39.7	42.1	47.8
O_{pre}	32.5	**45.3**	48.7	**51.2**	**54.3**	73.2	25.1	**30.1**	**38.1**	40.8	42.9	**45.5**	48.8
Δ_{pre}	−1.1	+1.3	−0.2	+0.3	+1.8	+7.1	+7.4	+5.4	+6.3	+2.8	+3.2	+3.3	+1.0
Host: Faster R-CNN (ResNet-18-FPN)													
B_{rnd}	0.5	1.7	2.7	3.8	5.1	28.6	5.5	10.3	17.5	19.2	21.9	25.6	36.4
O_{rnd}	16.7	28.9	33.3	36.3	39.4	56.3	21.8	25.4	**35.2**	37.0	**39.3**	39.0	41.5
Δ_{rnd}	+16.2	+27.2	+30.6	+32.5	+34.3	+27.7	+16.3	+15.1	+17.8	+17.8	+17.5	+13.4	+5.1
B_{pre}	5.2	15.1	18.5	21.9	24.5	44.0	13.4	19.3	28.1	33.7	35.9	37.9	**44.7**
O_{pre}	**18.3**	**31.3**	**36.0**	**39.0**	**41.9**	**57.8**	**22.6**	**26.4**	34.1	**37.3**	37.9	**40.6**	42.5
Δ_{pre}	+13.1	+16.2	+17.5	+17.1	+17.4	+13.8	+9.1	+7.1	+6.1	+3.6	+2.0	+2.7	−2.1

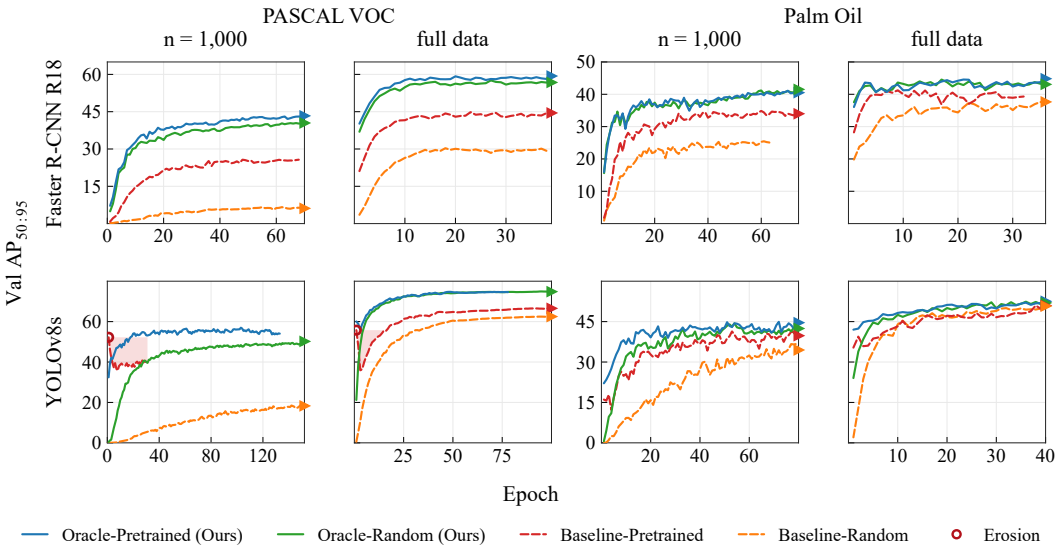

Figure 3: **The prior accelerates convergence on both hosts and datasets.** Validation curves of one representative partition at $n = 1{,}000$ and at full data; Appendix K shows the other budgets. The circle marks the pretrained YOLOv8s baseline's early peak on PASCAL VOC, and shading the accuracy it loses afterwards (Section 5).

error falls by 32% and 50%, and false-negative error by 49% and 58%. Qualitative feature maps of the Faster R-CNN pair support this, as the fused C5 map follows the object-level response of the frozen encoder, whereas the baseline responds only around the query point (Appendix M). On Palm Oil, classification is the largest error type. The prior barely changes it on YOLOv8s, whose gain coincides instead with lower false-negative error, and raises it by 71% on Faster R-CNN, in line with that host's deficit at full data. One possible explanation is that the prior becomes redundant on Palm Oil, for either of two reasons. With plentiful labels, the host's own hypothesis class may already suffice to separate categories that differ mainly in local colour and texture, which corresponds to the

Table 2: **TIDE error decomposition at full data.** Pretrained baseline → hybrid at IoU 0.50; error columns are AP_{50} points, lower is better. Cls, classification; Loc, localisation; Bkg, background; Miss, missed ground truth; FP and FN, all false positives and all false negatives. Bold marks a change of at least 1 point and 25% of the baseline value.

Host · dataset	AP_{50}	Cls	Loc	Bkg	Miss	FP	FN
YOLOv8s · PASCAL VOC	84.5→90.3	**2.4→1.1**	2.0→1.4	3.9→3.2	0.9→0.4	**10.8→7.4**	**3.6→1.8**
YOLOv8s · Palm Oil	67.5→71.2	21.5→20.9	2.2→2.6	1.2→1.1	0.7→0.7	13.5→13.2	15.4→12.9
FRCNN-R18 · PASCAL VOC	73.1→87.2	**2.3→0.5**	3.9→3.0	**4.4→3.1**	1.2→0.4	**20.1→10.1**	**4.7→2.0**
FRCNN-R18 · Palm Oil	77.7→72.6	**7.9→13.5**	3.5→3.8	**1.8→0.8**	0.7→1.2	18.2→20.1	**3.1→5.3**

Table 3: **Full-scale COCO.** $AP_{50:95}$ (AP) and its standard breakdowns on `val2017`, one run per condition (Section 4.3); AR_{100} is average recall at 100 detections per image. Params (M) gives total and trainable parameters in millions; the oracle's frozen remainder is the 21.60 M DINOv3 ViT-S/16 encoder.

Configuration	Params (M) total / trainable	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_{100}
Baseline (Faster R-CNN R50)	41.7 / 41.7	8.9	23.5	5.2	1.2	2.9	9.9	31.6
Oracle (+ frozen ViT)	72.5 / 50.9	11.3	29.5	6.5	1.8	4.5	12.5	36.4
Δ Oracle − Baseline	30.8 / 9.2	+2.3	+6.0	+1.4	+0.6	+1.6	+2.6	+4.7

estimation-limited reading of H2 (Appendix F); alternatively, the task may not require the scene-level information that the prior supplies. On Faster R-CNN, the redundant prior appears to interfere with the classification decision.

4.3 Scalability to COCO

We test whether the gain transfers to a larger, 80-class taxonomy by training Faster R-CNN with a ResNet-50 host on the full COCO training set (Table 3). This is a restricted scalability test, in which each condition is trained once, for 12 epochs at 512 px,[1] so both detectors remain far from convergence and are compared only with each other. Under this budget, the prior adds 2.3 AP, a 26% relative improvement, raises AP for small, medium and large objects, and improves 78 of the 80 categories.

4.4 Ablations and computational cost

Unless noted otherwise, ablations train the Faster R-CNN ResNet-18-FPN host for 24 epochs at 640×640 on the three disjoint $n = 1{,}000$ VOC partitions and report mean test $AP_{50:95}$. Each ablation varies one design decision and keeps the others at the reference configuration. Results vary by about 0.5 AP between partitions, so smaller differences are not read as effects. Table 4 covers the three structural decisions of the design; the remaining choices are reported in Appendix B.

Injection site. We investigate whether the prior should enter at a single deep stage or across the feature pyramid by evaluating every non-empty subset of the backbone stages C3, C4 and C5, at strides 8, 16 and 32 (Table 4a). The token grid is max-pooled at C5, used directly at C4 and upsampled at C3, where a learned 3×3 convolution follows (Appendix G); the added parameters grow with the stage's width and this convolution. On the pretrained host every placement adds at least 12 points over the baseline, so the presence of the prior matters more than its position. The best position depends on initialisation. On the pretrained host, accuracy rises with depth and no combination improves on C5 alone; on the randomly initialised host, every combination of two or three stages outperforms every single stage. This is likely because shallow coupling perturbs the

[1] Training a new architecture on COCO under a standard schedule, and repeating each run, exceeded our compute budget. The experiment therefore tests whether the gain transfers to an 80-class taxonomy, not the accuracy either detector can reach.

Table 4: **Ablations of the injection site, host capacity and integration route.** Test $AP_{50:95}$; parameters in millions; bold marks the best AP in each column. **(a)** ResNet-18 host; *Added*, trainable parameters of the interface; Pre./Rnd., pretrained/randomly initialised host. **(b)** Pretrained ResNet-50 and WRN-50-2 hosts, full PASCAL VOC, 36 epochs, one run per cell. **(c)** 480 px detector input. The encoder serves as the backbone behind a simple feature pyramid of w channels (SFP-w), fine-tuned (ViTDet) or frozen; the Oracle is the ResNet-18 host with SGDL. Params, total / trainable; Train s/ep, compute-only time per epoch; Train GiB, peak training memory; Δ AP against the Oracle. † reference configuration. ‡ not run.

Site	Added (M)	Pre.	Rnd.
Baseline	–	23.1	– ‡
C3	0.23	35.7	35.5
C4	0.23	37.3	36.8
C5	0.72	**39.2**	36.3
C3+C4	0.46	36.4	37.3
C3+C5	0.95	38.6	37.4
C4+C5	0.95	38.9	37.9
C3+C4+C5	1.18	38.2	**38.2**

(a) Injection site

Backbone	Params (M)	AP
ResNet-50	41.4	50.0
+ ViT	72.2	60.6
WRN-50-2	84.7	50.0
+ ViT	115.5	**60.8**

(b) Host capacity

Route	Params (M) total / trainable	Train s/ep	Train GiB	AP	Δ AP
Oracle (ours)†	50.7 / 29.1	52.3	1.12	38.1	–
ViTDet, SFP-256	39.5 / 39.5	63.2	2.10	**41.8**	+3.8
Frozen, SFP-256	39.5 / 17.9	52.1	1.15	35.0	−3.0
Frozen, SFP-368	48.4 / 26.8	55.9	1.57	35.1	−3.0
Frozen, SFP-448	55.4 / 33.8	58.9	1.89	35.1	−3.0

(c) Integration route

competent early features of a pretrained host, whereas a randomly initialised host has not yet learned them. Since the pretrained host is the practical deployment, SGDL injects once, at C5.

Host capacity. To test whether the gain is simply added capacity, we widen a pretrained ResNet-50 host to WRN-50-2 (Zagoruyko & Komodakis, 2016) instead,[2] training on full PASCAL VOC for 36 epochs with one run per cell (Table 4b). Doubling the width of each bottleneck block adds 43.3 M trainable parameters and leaves accuracy unchanged, whereas SGDL adds 9.2 M trainable parameters and a frozen encoder and raises accuracy by about ten points. Added to the wider host, the prior raises accuracy by the same margin again. The gain is therefore unlikely to stem from capacity; the prior appears to supply information that a wider convolutional host does not acquire, which supports H1 (Appendix F).

Integration route. We next investigate whether the encoder would serve better as the detector's backbone than as a supplier to a convolutional host (Table 4c, at a 480 px detector input). We compare SGDL on the ResNet-18 host with two routes that place the encoder behind the simple feature pyramid (SFP) of ViTDet (Li et al., 2022b), which builds multi-scale maps from its single-scale output. Kept frozen, the encoder-as-backbone trails SGDL by 3.0 AP at every pyramid width from 256 to 448 channels, which suggests that the deficit lies in the missing convolutional hierarchy rather than in pyramid width. Trained end to end, as in ViTDet, the encoder overtakes SGDL by 3.8 AP, but it trains every encoder weight and needs 1.9 times the training memory; fine-tuning the encoder inside SGDL yields a gain of similar size (Appendix B). We keep it frozen for lower cost, reuse across detectors and an identical prior in every comparison.

Computational cost. The frozen branch is cheap in trainable parameters but expensive in compute. On YOLOv8s it adds only 0.72 M trainable parameters, as on the ResNet-18 host, but 58.8 GMACs, which raises its compute to $5.1\times$ that of the baseline and lowers throughput from 102.6 to 31.3 frames per second (Appendix D). Almost all of this compute comes from the encoder, not the interface, and grows with its number of patch tokens, so its input resolution is the main lever on cost. Feeding the encoder 480 px instead of 640 px roughly halves the added compute and keeps 91% of the gain (Table 7).

5 Refractory feature erosion during adaptation

The COCO-pretrained YOLOv8s starts fine-tuning on PASCAL VOC already able to detect every target category, since all twenty lie inside the COCO label space. Within the first few epochs, how-

[2] Randomly initialised hosts reach lower accuracy and converge more slowly, so we ask the capacity question of competent, pretrained detectors.

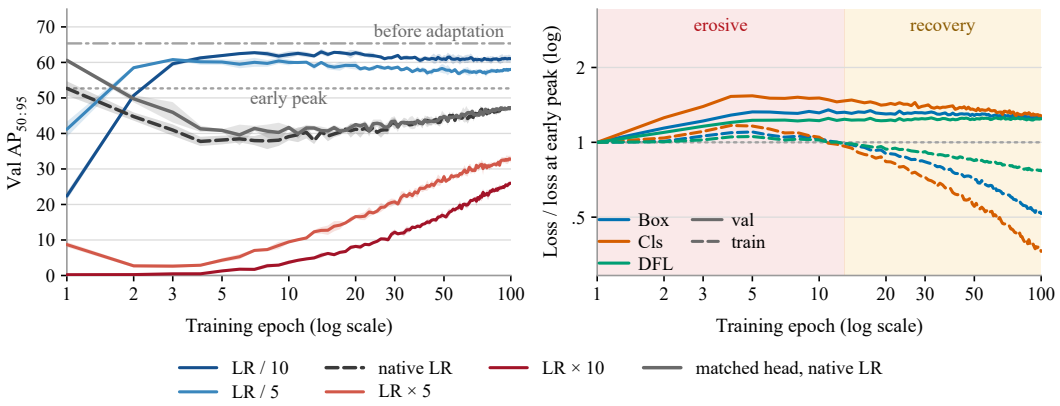

Figure 4: **Lowering the learning rate removes the erosion of YOLOv8s.** COCO-pretrained YOLOv8s fine-tuned on PASCAL VOC at $n = 1{,}000$; curves are means over three disjoint partitions. **Left:** Validation $AP_{50:95}$ under the learning-rate factors shown; bands are ± 1 s.d.; the dash-dot line marks the AP before adaptation and the dotted line the early peak, the highest native-rate AP within the first five epochs. **Right:** Losses of the matched-head run at the native rate, divided by their early-peak values; red shading marks the erosive phase and amber the recovery. Faster R-CNN and YOLOv6-S are shown in Figure 5.

ever, it loses much of that accuracy. We call this trajectory *refractory feature erosion*. It resembles the stability gap of continual learning, a transient loss of earlier competence when training on new data begins (Lange et al., 2023), but here the target categories are already known to the detector. It appears at every VOC budget (Figure 3; Appendix K), whereas the hybrid, trained with the same learning rate, shows no comparable collapse.

To identify the cause, we fine-tune COCO-pretrained detectors on $n = 1{,}000$ VOC images over three disjoint partitions for 100 epochs and test four explanations (Figure 4; Appendix A). The official YOLOv6-S implementation (Li et al., 2022a) erodes in the same way, which rules out an implementation error. We next test whether the rebuilt 20-class classifier head is responsible, since a randomly initialised head can disturb pretrained features during fine-tuning (Kumar et al., 2022). A head initialised with the twenty matching COCO classifier rows erodes at least as much, so the head is not the cause. We then scale the native learning rate by factors from $1/10$ to 10. Lowering it removes the collapse, so the learning rate causes the erosion we observe. Training and validation losses also rise together while AP falls, which is inconsistent with ordinary overfitting. Finally, a COCO-pretrained Faster R-CNN adapted in the same way overfits instead, so the signature appears in both YOLO detectors but not in Faster R-CNN.

6 Conclusion

We introduced SGDL, a single-site interface that fuses a frozen ViT into a trainable convolutional detector without changing its neck, heads or loss, and used it to study how a frozen global prior interacts with a local detector. Across one- and two-stage hosts, the prior accelerates learning and substitutes for much of conventional host pretraining, to the point that a randomly initialised hybrid can surpass a pretrained detector. Its benefit is not uniform, however. It persists with abundant labels where scene context aids recognition and contracts where local appearance suffices, so the value of a global prior depends on what the target task requires. We also identified refractory feature erosion, in which a pretrained YOLO detector fine-tuned at its default learning rate loses accuracy on categories it already detects. The main limitations are the inference compute of the frozen encoder, which a lower encoder resolution reduces substantially, and the scope of the evidence, which covers two target domains and one shortened COCO run per condition. Explaining erosion and testing further domains, detectors and encoders are natural next steps. We hope that SGDL serves both as a practical way to add frozen ViT priors to existing detectors and as a controlled instrument for studying when they help.

AI USE STATEMENT

In accordance with ICLR guidelines, we disclose the use of generative AI tools to assist with execution, drafting, and manuscript polishing. All research questions, hypotheses, experimental designs, literature reviews, and scientific conclusions were conceived and developed entirely by the authors; AI was not used for idea generation.

For execution and codebase development, AI tools were utilized to refine author-written Python code, improving robustness of the written code. AI also assisted in writing scripts for quick hypothesis testing (for example, prototyping scripts to evaluate learning-rate feature erosion). To handle the experimental data, which was distributed across numerous CSV files, AI tools helped draft the Python parsers used for data aggregation. The AI did not perform the data analysis or aggregation itself; it only assisted in writing the parsing logic, which the authors then executed. All AI-assisted code and parser outputs were manually checked, tested, and verified by the authors.

For drafting and polishing, AI was used to refine the manuscript's text and ensure clarity. A specific focus was placed on formatting the mathematical notation in LaTeX to ensure strict adherence to the newly added ICLR conventions. No complete draft of the paper, experimental data, or datasets were generated by AI. The authors thoroughly reviewed all AI-assisted code, text, analyses, and formatting, and assume full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies object-detection methodology using existing research datasets and does not involve the collection of human-subject data by the authors. We do not introduce a new dataset or deploy the proposed system in a high-stakes decision-making setting. We are not aware of ethical concerns specific to the methodology beyond the general limitations and potential misuse associated with computer-vision systems.

REPRODUCIBILITY STATEMENT

Appendices A and B describe the erosion and ablation protocols, Appendix D the cost measurements, Appendix G the SGDL interface, Appendix H the datasets and the stratified partitions, Appendix I every training and evaluation setting, and Appendix J the between-partition variability. The complete codebase will be released publicly when the conference anonymity requirements permit. This release will include the training and evaluation code, all reported ablation experiments, and the custom stratification utility used to construct the controlled data regimes.

REFERENCES

Jean-Yves Audibert and Alexandre B. Tsybakov. Fast learning rates for plug-in classifiers. *The Annals of Statistics*, 35(2):608–633, 2007.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Josh Beal, Eric Kim, Eric Tzeng, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Toward Transformer-Based Object Detection. *CoRR*, abs/2012.09958, 2020.

Irwan Bello, Barret Zoph, Quoc Le, Ashish Vaswani, and Jonathon Shlens. Attention Augmented Convolutional Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3285–3294, 2019. doi: 10.1109/ICCV.2019.00338.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-yuan Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. Tide: A general toolbox for identifying object detection errors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 558–573, Cham, 2020. Springer International Publishing.

Léon Bottou and Olivier Bousquet. The Tradeoffs of Large Scale Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 161–168. Curran Associates, Inc., 2007.

Zhi Cai, Yingjie Gao, Yanan Zhang, Xinzhu Ma, and Di Huang. Towards unbiased source-free object detection via vision foundation models. *Pattern Recognition*, 179:113545, 2026.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 213–229, Berlin, Heidelberg, 2020. Springer-Verlag. doi: 10.1007/978-3-030-58452-8_13.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-Former: Bridging MobileNet and Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5260–5269, Los Alamitos, CA, USA, 2022. IEEE Computer Society. doi: 10.1109/CVPR52688.2022.00520.

Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision Transformer Adapter for Dense Predictions. In *International Conference on Learning Representations (ICLR)*, 2023.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations (ICLR)*, volume 2024, pp. 2632–2652, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Yann Dubois, Tatsunori Hashimoto, and Percy Liang. Evaluating Self-Supervised Learning via Risk Decomposition. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8779–8820. PMLR, 2023.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. doi: https://doi.org/10.1016/j.neunet.2017.12.012.

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. doi: 10.1007/s11263-009-0275-4.

Shenghao Fu, Junkai Yan, Qize Yang, Xihan Wei, Xiaohua Xie, and Wei-Shi Zheng. Frozen-DETR: Enhancing DETR with Image Understanding from Frozen Foundation Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Xiangyu Gao, Yu Dai, Benliu Qiu, Lanxiao Wang, Heqian Qiu, and Hongliang Li. Modulating CNN features with pre-trained ViT representations for open-vocabulary object detection. arXiv preprint arXiv:2501.16981, 2025.

Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. LeViT: A Vision Transformer in ConvNet's Clothing for Faster Inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12259–12269, 2021.

Yunusa Haruna, Shiyin Qin, Abdulrahman Hamman Adama Chukkol, Abdulganiyu Abdu Yusuf, Isah Bello, and Adamu Lawan. Exploring the synergies of hybrid convolutional neural network and vision transformer architectures for computer vision: A survey. *Engineering Applications of Artificial Intelligence*, 144:110057, 2025. doi: https://doi.org/10.1016/j.engappai.2025.110057.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Kaiming He, Ross B. Girshick, and Piotr Dollár. Rethinking ImageNet Pre-Training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4917–4926. IEEE, 2019.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked Autoencoders Are Scalable Vision Learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988. IEEE, 2022.

Shihua Huang, Yongjie Hou, Longfei Liu, Xuanlong Yu, and Xi Shen. Real-Time Object Detection Meets DINOv3. *CoRR*, abs/2509.20787, 2025.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 2015. PMLR.

Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024. URL `https://github.com/ultralytics/ultralytics`. Version 11.0.0.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023. URL `https://github.com/ultralytics/ultralytics`. Version 8.0.0.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution. In *International Conference on Learning Representations (ICLR)*, 2022.

Weicheng Kuo, Yin Cui, Xiuye Gu, A. J. Piergiovanni, and Anelia Angelova. Open-Vocabulary Object Detection upon Frozen Vision and Language Models. In *International Conference on Learning Representations (ICLR)*, 2023.

Matthias De Lange, Gido M van de Ven, and Tinne Tuytelaars. Continual evaluation for lifelong learning: Identifying the stability gap. In *International Conference on Learning Representations (ICLR)*, 2023.

Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. YOLOv6: A single-stage object detection framework for industrial applications. *ArXiv*, abs/2209.02976, 2022a. URL `https://api.semanticscholar.org/CorpusID:252110986`.

Yanghao Li, Hanzi Mao, Ross B. Girshick, and Kaiming He. Exploring Plain Vision Transformer Backbones for Object Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pp. 280–296. Springer, 2022b.

J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi: 10.1109/18.61115.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 740–755, Cham, 2014. Springer International Publishing.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017. doi: 10.1109/CVPR.2017.106.

Yutong Lin, Ze Liu, Zheng Zhang, Han Hu, Nanning Zheng, Stephen Lin, and Yue Cao. Could Giant Pre-trained Image Models Extract Universal Representations? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

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 (ICCV)*, pp. 9992–10002, 2021. doi: 10.1109/ICCV48922.2021.00986.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29. Curran Associates, Inc., 2016.

P. Malaisree, S. Youwai, T. Kitkobsin, S. Janrungautai, D. Amorndechaphon, and P. Rojanavasu. DINO-YOLO: Self-supervised pre-training for data-efficient object detection in civil engineering applications. arXiv preprint arXiv:2510.25140, 2025. URL `https://arxiv.org/abs/2510.25140`.

Soichiro Okazaki, Tatsuya Sasaki, and Hiroki Ohashi. DetRefiner: Model-Agnostic Detection Refinement with Feature Fusion Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 6890–6900, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Namuk Park and Songkuk Kim. How Do Vision Transformers Work? In *International Conference on Learning Representations (ICLR)*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Zhiliang Peng, Zonghao Guo, Wei Huang, Yaowei Wang, Lingxi Xie, Jianbin Jiao, Qi Tian, and Qixiang Ye. Conformer: Local Features Coupling Global Representations for Recognition and Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9454–9468, 2023. doi: 10.1109/TPAMI.2023.3243048.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pp. 8748–8763. PMLR, 2021.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do Vision Transformers See Like Convolutional Neural Networks? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788. IEEE Computer Society, 2016.

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(06):1137–1149, 2017. doi: 10.1109/TPAMI.2016.2577031.

Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pp. 145–158, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel HAZIZA, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Herve Jegou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Transactions on Machine Learning Research*, 2026.

Jaskirat Singh, Xingjian Leng, Zongze Wu, Liang Zheng, Richard Zhang, Eli Shechtman, and Saining Xie. What matters for representation alignment: Global information or spatial structure? In *International Conference on Learning Representations (ICLR)*, 2026.

Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck Transformers for Visual Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16519–16529, 2021.

Suharjito, Martinus Grady Naftali, Gregory Hugo, Muhammad Reza Azhar Priyadi, Muhammad Asrol, and Ditdit Nugeraha Utama. Oil palm fruits dataset in plantations for harvest estimation using digital census and smartphone. *Scientific Data*, 12(1):972, 2025. doi: 10.1038/s41597-025-05227-x.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pp. 10347–10357. PMLR, 2021.

Alexander B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135 – 166, 2004. doi: 10.1214/aos/1079120131.

Ultralytics. MuSGD optimizer, ultralytics API reference (`ultralytics/optim/muon.py`). `https://docs.ultralytics.com/reference/optim/muon`, 2025. Accessed 2026-09-25.

Cristina Nader Vasconcelos, Vighnesh Birodkar, and Vincent Dumoulin. Proper Reuse of Image Classification Features Improves Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13618–13627. IEEE, 2022.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 548–558, 2021. doi: 10.1109/ICCV48922.2021.00061.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-Local Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Xin Wang, Thomas E. Huang, Joseph Gonzalez, Trevor Darrell, and Fisher Yu. Frustratingly Simple Few-Shot Object Detection. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9919–9928. PMLR, 2020.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing Convolutions to Vision Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22–31. IEEE, 2021.

Size Wu, Wenwei Zhang, Lumin Xu, Sheng Jin, Xiangtai Li, Wentao Liu, and Chen Change Loy. CLIPSelf: Vision Transformer Distills Itself for Open-Vocabulary Dense Prediction. In *International Conference on Learning Representations (ICLR)*, 2024.

Chunlong Xia, Xinliang Wang, Feng Lv, Xin Hao, and Yifeng Shi. ViT-CoMer: Vision Transformer with Convolutional Multi-scale Feature Interaction for Dense Predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5493–5502, 2024. doi: 10.1109/CVPR52733.2024.00525.

Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross B. Girshick. Early Convolutions Help Transformers See Better. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 30392–30400, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1–87.12. BMVA Press, 2016. doi: 10.5244/C.30.87.

A Analysis of refractory feature erosion

A.1 The phenomenon

We identified the behaviour studied here by accident, during the stratified PASCAL VOC experiments of Section 4.2, in which every YOLOv8s run uses the learning rate that Ultralytics selects automatically. All twenty VOC categories lie inside the COCO label space, so the COCO-pretrained YOLOv8s baseline begins adaptation already able to detect every target category. We therefore expected it to keep most of this competence and, with few labels, to overfit slowly. Its validation AP instead fell sharply within the first few epochs of fine-tuning, although the learning rate was the framework's default for this model and not an aggressive setting. Preliminary trial runs with YOLO11s (Jocher & Qiu, 2024) showed the same trajectory.

A.2 Candidate explanations and experimental design

We consider four explanations, each of which predicts a different outcome.

1. **An implementation error in the Ultralytics training pipeline**, our first assumption. If it holds, an independent implementation of a YOLO detector should not erode.
2. **Reinitialisation of the classifier head.** Adapting YOLOv8s to VOC rebuilds its classification head without copying the corresponding COCO weights natively. However, all 20 VOC classes have counterparts in COCO, and the pretrained detector already detects them, making the sharp drop in VOC accuracy after fine-tuning surprising. Kumar et al. (2022) show, for out-of-distribution generalisation, that jointly fine-tuning pretrained features with a randomly initialised head can distort those features. If the rebuilt head is the main cause, copying the matching COCO classifier rows should substantially reduce the drop.
3. **The adaptation step size.** If the native learning rate is too large for this transfer, a smaller rate should remove the collapse.
4. **A general property of adapting a detector whose source label space contains the target.** If it holds, a detector of another family should erode as well.

We test these explanations on three COCO-pretrained detectors from two families: YOLOv8s; YOLOv6-S in its official implementation (Li et al., 2022a), which shares no training code with Ultralytics; and Faster R-CNN with a ResNet-50-FPN backbone, initialised from the COCO weights of torchvision and trained with the same optimiser and schedule as in our main experiments. Each detector is fine-tuned on the three disjoint $n = 1{,}000$ VOC partitions for 100 epochs with early stopping disabled, scored on the common validation split after every epoch, and otherwise run with its framework's own training pipeline (Table 5). We call the learning rate that each framework uses by default its native rate.[3]

The detectors differ in optimiser, schedule and augmentation, so we do not compare them setting by setting. Within each detector, the learning rate is the only variable we change. We multiply the native schedule by a constant factor and hold every other setting fixed, so each learning-rate ladder is controlled within its own detector. Because the detectors belong to different architectures, we start each one from its framework's default rate and vary the rate around it, rather than imposing a common value. The Faster R-CNN ladder, from 10^{-5} to 10^{-3}, nevertheless spans the native YOLOv8s rate of 4.17×10^{-4}, so the comparison between families also covers the rate at which YOLOv8s erodes. YOLOv6-S follows the same scheme around its own native SGD rate of 3.2×10^{-3}. The only other departure from the native pipelines is the *matched-head* control for YOLOv8s, in which the twenty COCO classifier rows that correspond to the VOC categories are copied into the 20-class head before adaptation; the native Ultralytics path copies none of them, and YOLOv6-S keeps its native path, which also copies none. Faster R-CNN has no native procedure for a new label set, and we copy the same twenty rows into its head in every run.

To show that each detector is competent before adaptation, we evaluate it once on the VOC validation split with the twenty corresponding classifier rows and no parameter update. YOLOv8s reaches

[3] Ultralytics selects the YOLOv8s rate automatically, and the official YOLOv6-S configuration fixes its own. torchvision defines no fine-tuning default for Faster R-CNN, whose reference recipe uses SGD; its native rate is the AdamW rate of our main experiments (Appendix I).

65.4 $AP_{50:95}$ and Faster R-CNN 55.6 AP. Adaptation with 1,000 images is not expected to reach these values; they serve as a reference. Because the detectors peak at different epochs, we describe each trajectory from its *early peak*, the highest native-rate validation AP within the first five epochs (epoch 1 for YOLOv8s, 2 for YOLOv6-S and 5 for Faster R-CNN).

A.3 RESULTS

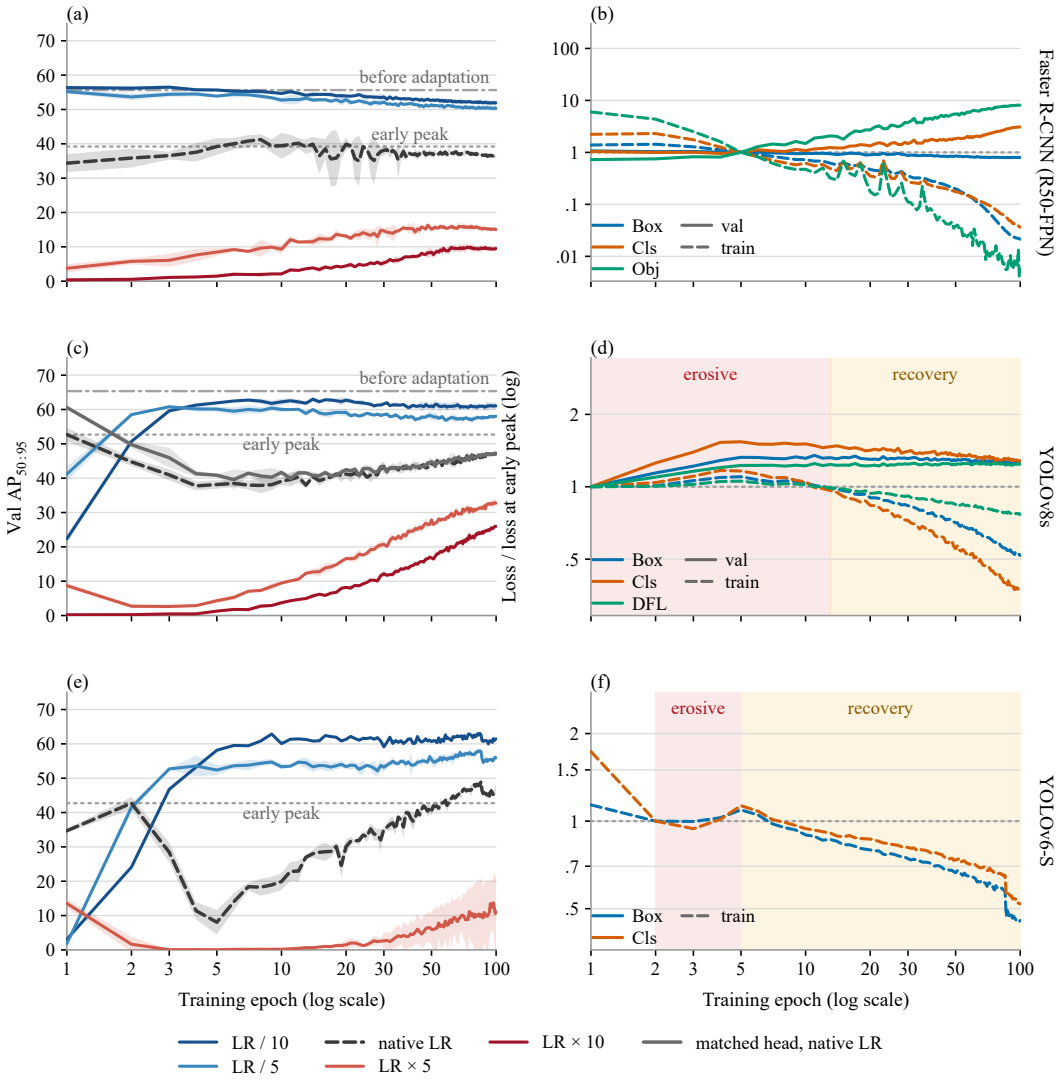

Figure 5: **Learning-rate intervention on Faster R-CNN, YOLOv8s and YOLOv6-S.** COCO-pretrained detectors fine-tuned on PASCAL VOC at $n = 1{,}000$; curves are means over three disjoint partitions unless Table 5 notes fewer. **(a,c,e)** Validation $AP_{50:95}$ with each host's native learning rate scaled by the factors shown; for YOLOv8s, the grey curve is the matched-head control at the native rate. Bands are ± 1 s.d.; dash-dot lines mark the AP before adaptation, which was not measured for YOLOv6-S, and dotted lines the early peak. **(b,d,f)** Training (dashed) and validation (solid) losses at the native rate, each divided by its value at the early peak; the YOLOv8s panel shows the matched-head run, and the YOLOv6-S panel shows training losses only, since its official trainer does not report validation losses. Red shading marks the erosive phase, from the early peak until the training losses return to their early-peak values, or for YOLOv6-S until its first AP minimum; amber marks the recovery that follows.

Table 5: **Detectors and learning rates in the erosion analysis.** Each detector is COCO-pretrained, fine-tuned at 640×640 on the three disjoint $n = 1{,}000$ PASCAL VOC partitions for 100 epochs, and otherwise run with its framework's own settings. *COCO classifier rows* is the number of the twenty VOC categories whose COCO classifier weights the head starts from. *Learning-rate factors* multiply the native schedule; within each detector, they are the only setting that changes.

	YOLOv8s	**YOLOv6-S**	**Faster R-CNN**
Implementation	Ultralytics 8.4.37	Official (Meituan)	torchvision, ResNet-50-FPN
COCO classifier rows	0/20; 20/20 (matched head)	0/20	20/20
Optimiser, native rate	AdamW, 4.17×10^{-4}	SGD, 3.2×10^{-3}	AdamW, 10^{-4}
Learning-rate factors	$1/10$, $1/5$, 1, 5, 10	$1/10^{*}$, $1/5$, 1, 5^{*}	$1/10$, $1/5$, 1, 5, 10
Validation loss	Logged	Not reported	Logged

*Completed on one ($1/10$) and two (5) of the three partitions; the remaining runs, and every run at $10\times$, failed through numerical instability. Curves at these factors average the completed partitions.

Implementation. We first test whether erosion is an artefact of the Ultralytics code by repeating the experiment in the official YOLOv6-S implementation, which erodes in the same way (Figure 5(e,f)).[4] Its validation AP falls from 42.7 at its early peak to 8.0 at epoch 5 and then recovers to 46.4 by epoch 100, and its training losses rise while AP falls. YOLOv6-S also trains with its native SGD optimiser, whereas YOLOv8s uses AdamW, so erosion is not tied to one optimiser. Surprisingly, it occurs even at the learning rate each framework selects by default.

Head initialisation. The matched head gives a better start, 60.6 AP after the first epoch against 52.7 for the rebuilt head, so the transferred classifier rows are useful. It nevertheless enters the same decline and loses more of its early-peak accuracy than the rebuilt head (Figure 5(c)), and both conditions end at 47.2 AP at epoch 100. A reinitialised head is therefore not required for erosion, and a well-initialised head does not prevent it.

Learning rate. At one fifth and one tenth of the native rate the sharp trough disappears in YOLOv8s (Figure 5(c)), and YOLOv6-S responds to its learning rate in the same way (Figure 5(e)). The learning rate is therefore a causal factor in the erosion we observe, since lowering it removes the collapse while every other setting is held fixed. Erosion also persists under a third optimiser. At full data, `optimizer=auto` selects MuSGD at 10^{-2} for YOLOv8s (Table 13), and validation AP still falls from 55.8 after the first epoch to 32.2 at epoch 3, while all three training losses rise above their first-epoch values, before it returns above its first-epoch level by epoch 15. At five and ten times the native rate, no detector reaches a competent state, and each remains far below its lower-rate runs throughout training. Such failure is expected under excessive step sizes and says little about erosion itself.

Loss dynamics. The training loss separates erosion from overfitting (Figure 5(b,d,f)). In both YOLO detectors, the training losses rise above their early-peak values while validation AP collapses and turn downward near the AP trough, where the recovery begins; in YOLOv8s, the validation losses rise with them. In YOLOv6-S, the box and classification losses both peak at epoch 5, the AP trough, at 1.09 and 1.13 times their early-peak values. The detector thus becomes worse on the data it is trained on. This is inconsistent with ordinary overfitting, in which training loss continues to fall while validation accuracy deteriorates. Faster R-CNN instead overfits. At its native rate it climbs from 34.4 AP after the first epoch to 41.2 AP by epoch 8 and then drifts down to 36.5 AP, while its training loss falls throughout and its validation classification and objectness losses rise. No arm of its learning-rate ladder, which spans the native YOLOv8s rate, shows the early peak, sharp trough and recovery of the YOLO detectors.

[4]The official YOLOv6-S trainer logs training losses only, so panel (f) has no validation curves. Its runs also became numerically unstable at both ends of the learning-rate ladder. The native and $1/5$ arms completed on all three partitions and are averaged like the other detectors, whereas the $1/10$ and $5\times$ curves rest on one and two partitions, and no $10\times$ run completed (Table 5).

A.4 IDENTIFYING THE EROSION SIGNATURE

Faster R-CNN thus adapts as expected when the source label space contains the target categories and overfits only slowly. The YOLO detectors instead follow a three-part trajectory, with a competent state early in adaptation, a sharp fall within a few epochs during which the training loss rises, and a recovery. We call this trajectory *refractory feature erosion*. *Erosion* refers to the loss of transferred competence, and *refractory* to the recovery that follows. The name describes the trajectory and does not assert a mechanism.

A.5 REPRODUCING THE EROSION

Erosion requires no part of our framework and can be reproduced with public checkpoints and stock training code. It suffices to fine-tune a COCO-pretrained YOLOv8s or YOLO11s from Ultralytics, or YOLOv6-S from its official repository, on PASCAL VOC with the framework's default settings, and to record validation AP after every epoch. The fall appears within the first few epochs at every training-set size we test, including the full training set, and is most evident when data are limited; scaling the learning rate down by five removes it. We observed the trajectory in all three models; for Ultralytics models, the default `optimizer=auto` selects the learning rate from the dataset size and epoch budget (Table 13), so the same default command produces it without any configuration.

A.6 CONCLUSION AND OPEN QUESTIONS

The experiments rule out an implementation error and a reinitialised head, and identify the adaptation learning rate as a causal factor, since within YOLOv8s and within YOLOv6-S a lower rate removes the collapse while every other setting is held fixed. The trajectory appears in every YOLO detector we examine, including the YOLO11s trial runs and under three optimisers, and in no arm of the Faster R-CNN ladder. The ladder does not show why YOLO detectors are susceptible, which parameters change during the collapse, or why Faster R-CNN overfits instead; we leave a mechanistic analysis, and whether other one-stage detectors share the susceptibility, to future work.

Practical implications. Fine-tuning a competent pretrained detector remains worthwhile. At full VOC, the adapted YOLOv8s recovers from erosion and exceeds its pre-adaptation level, as the full-data pretrained baseline of Table 1 reaches 67.0 validation AP against 65.4 before adaptation. The risk lies in the erosion itself, and it affects practitioners and researchers alike. With limited data, recovery can remain incomplete. At $n = 1{,}000$, the native-rate YOLOv8s run ends at 47.2 AP, below the 52.7 AP it reached after its first epoch. A practitioner who keeps the final weights therefore receives a detector worse than an early checkpoint of the same run, and one who keeps the best checkpoint receives a detector that has barely adapted; the default training output signals neither. For example, a practitioner who fine-tunes a COCO-pretrained YOLO detector on a small traffic-camera dataset, whose vehicles and pedestrians the pretrained detector already detects well, may obtain a detector less accurate than the pretrained one. Without the per-epoch validation curve, nothing indicates that an earlier checkpoint or a lower learning rate would have given a better detector. Early stopping adds a further risk. Most training pipelines stop once validation AP has not improved for a fixed number of epochs, and after an early peak this patience criterion can end training during the collapse or the recovery, returning the pre-collapse checkpoint. For researchers, erosion can distort comparisons. A pretrained YOLO baseline evaluated at a fixed epoch budget may be measured during its recovery, and checkpoint selection by validation AP can return a checkpoint from before the collapse, as for the starred cells of Table 1.

Conditions for erosion. Every erosion we observed, in COCO-to-VOC adaptation with Palm Oil as the contrasting target, arose under the same two conditions, and a third governed how far the detector recovered.

1. **The source label space subsumes the target categories**, so the detector is competent before adaptation and has competence to lose. Palm Oil, whose five categories lie outside the COCO label space, provides the contrasting case. The same COCO-pretrained YOLOv8s starts there with little competence (8.8 to 16.1 AP after the first epoch at $n = 1{,}000$) and ends every run far above its early accuracy (Figure 3; Appendix K). This suggests that erosion requires the condition.

2. **The adaptation uses the framework's default learning rate.** The learning-rate ladder supports this condition directly, since lower rates remove the collapse.
3. **Limited target data restricts recovery.** This condition governs how much of the loss persists, not whether erosion occurs. Across the VOC budgets, recovery is partial at restricted budgets and complete at full data (Figure 3; Appendix K).

Testing these conditions on other domains and detectors would show whether they are sufficient and how often erosion affects fine-tuning in practice.

B ADDITIONAL ABLATIONS

This appendix reports the design choices outside the three structural decisions of Section 4.4: how the encoder grid is resampled onto C5, the resolution the encoder receives, whether the encoder is fine-tuned, and which frozen encoder provides the prior. All four follow the ablation protocol of Section 4.4 with the pretrained host and DINOv3 ViT-S/16 unless stated otherwise. The randomly initialised host converges slowly, so it is run only where its result informs a design choice, here the resampler. Tables 6, 7 and 8 report test $AP_{50:95}$ averaged over the three partitions.

Table 6: **Spatial resampler.** Test $AP_{50:95}$ with pretrained and randomly initialised hosts; *Added params* (M) counts the resampler's own parameters. Bold marks the best value in each column. † reference configuration.

Operator	Added params	Pretrained	Random
Max 2×2†	0.00	39.2	36.3
Avg 2×2	0.00	38.7	36.0
Conv 3×3	2.36	39.6	**37.2**
Conv 3×3 ×2	4.72	**39.9**	37.1

Spatial resampler. At a 640 px input the encoder returns a 40×40 grid of patch tokens (stride 16), whereas C5 is a 20×20 map (stride 32), so the projected encoder grid must be halved before the two are concatenated. We compare four operators for this step and change nothing else, so the projection and fusion blocks, each a 1×1 Conv–BN–SiLU, are identical to those of the reference interface. The four operators are 2×2 max-pooling and 2×2 average pooling with stride 2, which have no parameters; a single 3×3 convolution with stride 2; and a 3×3 convolution with stride 1 followed by a 3×3 convolution with stride 2, which first mixes neighbouring tokens at full resolution and then downsamples. Each convolution keeps the 512 channels of the projected grid and is followed by BatchNorm and SiLU. The learned operators improve on max-pooling by at most 0.9 AP on either host, and average pooling trails it by at most 0.5 AP (Table 6), differences close to the run-to-run variation of about 0.5 AP. One possible explanation for the small gap between parameter-free pooling and learned convolutions is that the prior mainly corrects classification errors (Section 4.2). Recognising what an object is likely depends more on whether a strong semantic activation reaches C5 than on its exact position within a 2×2 window, and max-pooling passes the strongest response in each window. This would also explain why max-pooling outperforms average pooling, which dilutes that response with its neighbours. We keep max-pooling because these gains are small and a learned resampler grows quadratically with the host's channel width, so that a single 3×3 convolution at the 2,048 channels of a ResNet-50 C5 would add about 38 M parameters. Where the parameter budget allows, a learned resampler is a reasonable alternative.

Encoder input resolution. Panel (a) of Table 7 varies the resolution fed to the frozen encoder while the detector stays at 640 px. The pooled encoder grid then no longer matches C5: at 480 px the encoder returns a 30×30 grid, which max-pooling reduces to 15×15. We keep the max-pool at every resolution and resize its output bilinearly to the 20×20 C5 grid, rather than removing the max-pool and resizing the encoder grid directly. Preliminary experiments favoured this order, with the max-pool helping even when its output was afterwards enlarged, and it keeps the operator identical across the ladder so that only the input resolution varies. Accuracy falls much more slowly than compute. At 480 px the prior retains 91% of its gain for 27.4 added GMACs, against 58.8 at 640 px, and at 256 px it retains 64% for 6.5 GMACs, about a ninth of the full overhead. Most of the gain disappears only at 112 px, where the encoder returns a 7×7 grid. Because the two streams take separate inputs, the encoder resolution can be lowered for deployment without changing the detector's own resolution.

Frozen versus fine-tuned encoder. We measure what freezing costs by fine-tuning the encoder together with the host at two learning rates (Table 7b). Fine-tuning raises AP from 39.2 to 43.0 at 10^{-5} and to 42.9 at 10^{-4}. A gain of this size is expected, since fine-tuning adapts a further 21.6 M parameters to the target domain. It brings two costs. Peak training memory grows by 1.65

Table 7: **Encoder input resolution and fine-tuning.** Test $AP_{50:95}$ with the pretrained host. **(a)** Encoder input resolution, with the detector at 640 px; *Patch tokens* is the number of tokens the encoder returns, *Added GMACs* the compute added over the no-ViT baseline and *Gain kept* the fraction of the 640 px gain retained. **(b)** Frozen or fine-tuned (FT) encoder; *Trainable* (M) counts the encoder's trainable parameters, *s/ep* is wall-clock time per epoch including validation (epochs 2–24), and *Train GiB* is peak `max_memory_allocated` during training. Bold marks the highest AP in each panel. † reference configuration.

Encoder input (px)	Patch tokens	Added GMACs	AP	Gain kept
Baseline	0	0	23.1	–
640†	1,600	58.8	**39.2**	100%
512	1,024	32.3	37.8	91%
480	900	27.4	37.7	91%
336	441	11.7	35.3	76%
256	256	6.5	33.4	64%
112	49	1.4	26.0	18%

(a) Encoder input resolution

Encoder	Trainable (M)	s/ep	Train GiB	AP
Frozen†	0.0	185.5	5.69	39.2
FT 10^{-5}	21.6	192.9	7.34	**43.0**
FT 10^{-4}	21.6	193.4	7.80	42.9

(b) Frozen vs fine-tuned

and 2.11 GiB, because the encoder's parameters now need gradients, optimiser states and stored activations. The adapted encoder also becomes specific to one dataset, whereas a frozen encoder is the same network in every detector and can be shared across them. We keep the encoder frozen in the main experiments for these reasons, and because freezing holds the prior identical in every matched comparison. When memory allows, fine-tuning at a small learning rate is a useful addition.

Table 8: **Choice of frozen encoder.** Measures of each encoder's frozen patch tokens (Appendix C) and test $AP_{50:95}$ of the hybrid detector trained with it, with Δ AP against the no-ViT baseline. -S and -B denote ViT-S/16 and ViT-B/16, and *Params* counts the frozen encoder. *Comp.* is the composite $z(\mathrm{LDS}) + z(-\mathrm{RMSC})$; *GAP* and *Region* are the macro-AP of the image-level and box-level linear probes. Values are means over three disjoint $n = 1{,}000$ partitions. Arrows mark higher-is-better columns; bold marks the best value in each. † reference configuration.

		Measured on frozen patch tokens					Trained hybrid detector	
Encoder	Params (M)	LDS	RMSC	Comp.	GAP AP↑	Region AP↑	AP↑	Δ AP
Baseline (no ViT)	–	–	–	–	–	–	17.8	–
DINOv3-S†	21.6	0.122	0.625	+3.55	**84.8**	**94.3**	**30.2**	**+12.4**
MAE-B	85.8	0.078	0.860	−0.01	74.1	86.4	26.7	+8.9
DeiT-S	21.7	0.080	0.896	−0.29	80.7	85.3	25.3	+7.5
DINO-B	85.8	0.032	0.861	−1.43	72.9	81.5	23.6	+5.8
DINO-S	21.7	0.038	0.922	−1.82	62.6	74.0	22.0	+4.1

Encoder choice. The main experiments use a single encoder, so we investigate how much the choice of frozen encoder matters by replacing DINOv3 ViT-S/16 with four other frozen ViTs (Table 8). Each encoder is attached at C5 of the same pretrained Faster R-CNN ResNet-18-FPN host, and the interface changes only in the input width of its first projection, which follows the encoder's token width. We assess each encoder over the three $n = 1{,}000$ partitions in two ways: by training the hybrid detector and measuring its test AP, and by measuring its frozen patch tokens without training a detector. The token measurements comprise two linear probes and two label-free statistics of the token grid, *local versus distant similarity* (LDS) and *RMS spatial contrast* (RMSC) (Singh et al., 2026); Appendix C defines and analyses all four.

The encoders vary two factors. The first is the pretraining objective: DINO learns by self-distillation (Caron et al., 2021), MAE by reconstructing masked pixels (He et al., 2022), and DeiT by supervised classification (Touvron et al., 2021), all on ImageNet-1K, whereas DINOv3 combines self-distillation with Gram anchoring and is trained on a much larger web-scale corpus (Siméoni et al.,

2026). The second is model capacity: DINO is tested as ViT-S/16 (21.7 M parameters) and ViT-B/16 (85.8 M), and MAE as ViT-B/16. All encoders receive 224 px inputs, the resolution at which most of them were pretrained; at 640 px their position embeddings would have to be interpolated far beyond the pretraining range, which could shift the distribution of their features. The detector also runs at 224 px, so the pooled 7×7 encoder grid matches $\mathrm{C5}$ exactly, and the no-ViT baseline is measured at the same resolution.

Every encoder improves the detector, by 4.1 to 12.4 AP, so SGDL works with frozen ViTs of different sizes and pretraining objectives without architecture-specific changes. DINOv3 gives the largest gain and DINO-S the smallest. Capacity does not decide the ranking. Moving DINO from ViT-S to ViT-B, with about four times the parameters, adds 1.7 AP, whereas at the ViT-S size the objective moves AP further, as DeiT-S exceeds DINO-S by 3.4 AP and DINOv3-S exceeds DeiT-S by 4.9 AP. MAE-B also exceeds the equally large DINO-B by 3.1 AP. The pretraining recipe therefore appears to matter more than capacity for this use. Capacity is also costly. A ViT-B/16 has about four times the parameters of a ViT-S/16 and runs about three times slower (292 against 940 images per second in Touvron et al. 2021), so a small encoder such as the ViT-S used throughout keeps the frozen branch affordable. Some pairs differ in more than one factor. DINOv3-S differs from DINO-S in both objective and training data, so their gap cannot be attributed to either alone. We hypothesise that what matters is how well the patch tokens preserve the spatial and category structure of the image, because the detector reads the token grid position by position rather than a pooled summary. The measurements on the frozen tokens are consistent with this hypothesis. The region probe and the LDS–RMSC composite both order the five encoders exactly as the detector does (Table 8; Appendix C).

C PATCH-PROBE DIAGNOSTICS FOR ENCODER CHOICE

The choice of frozen encoder changes detector AP by up to 8.3 points (Table 8), while each comparison requires a full detector training run for every encoder and partition. We therefore investigate whether properties of the frozen patch tokens alone anticipate the observed encoder ranking. We consider two label-free structural statistics, LDS and RMSC, together with two linear probes as label-based references. Because only five encoders are compared, all correlations are exploratory and are used to generate hypotheses rather than establish a predictive rule.

Why LDS and RMSC. SGDL injects the frozen patch-token grid into C5 position by position. The usefulness of an encoder may therefore depend not only on whether its tokens contain category information, but also on how that information is organised spatially. We use two statistics defined and evaluated by Singh et al. (2026): local versus distant similarity (LDS), which measures whether nearby patches are represented more similarly than distant patches, and RMS spatial contrast (RMSC), which measures the overall dispersion of patch-token directions within an image, that is, how homogeneous or heterogeneous the tokens are. They were evaluated for representation alignment in diffusion transformers; we use them to describe the frozen token grid and test whether they track detector AP across our encoders.

LDS and RMSC. For one image, let $\boldsymbol{x}_t \in \mathbb{R}^d$ denote the patch token at position t of an $H \times W$ grid, with $T = HW$, and define the cosine similarity

$$K(t, t') = \frac{\langle \boldsymbol{x}_t, \boldsymbol{x}_{t'} \rangle}{\|\boldsymbol{x}_t\|_2 \|\boldsymbol{x}_{t'}\|_2}. \tag{2}$$

Local versus distant similarity compares nearby and distant token pairs,

$$\mathrm{LDS} = \mathbb{E}[K(t, t') \mid 0 < \delta(t, t') < r] - \mathbb{E}[K(t, t') \mid \delta(t, t') \geq r], \tag{3}$$

where $\delta(t, t')$ is the distance between grid positions and $r = H/2$, equal to 7 for the 14×14 grid produced at 224 px. A larger LDS therefore means that nearby patches are, on average, more similar to one another than distant patches.

RMSC measures the dispersion of the normalised patch tokens. Let

$$\hat{\boldsymbol{x}}_t = \frac{\boldsymbol{x}_t}{\|\boldsymbol{x}_t\|_2}, \qquad \bar{\boldsymbol{x}} = \frac{1}{T} \sum_{t=1}^{T} \hat{\boldsymbol{x}}_t.$$

Then

$$\mathrm{RMSC} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} \|\hat{\boldsymbol{x}}_t - \bar{\boldsymbol{x}}\|_2^2}. \tag{4}$$

Because every $\hat{\boldsymbol{x}}_t$ has unit norm,

$$\mathrm{RMSC}^2 = 1 - \|\bar{\boldsymbol{x}}\|_2^2.$$

Thus, low RMSC means that patch-token directions are globally similar, whereas high RMSC means that they are more dispersed across the image. A strong scene-wide component is one possible source of low RMSC, but RMSC itself measures only this feature-space dispersion.

Both statistics are averaged over the 596 images of the common validation split and therefore depend only on the encoder. We follow Singh et al. (2026), except that δ is Euclidean distance on the patch grid rather than the Manhattan distance used in their definition.

Linear probes. As a label-based reference, we fit two linear classifiers to the frozen patch tokens for each training partition, discarding prefix tokens as SGDL does. The *GAP probe* averages all tokens inside the image content and predicts which of the 20 VOC classes are present. The *region probe* averages tokens whose centres fall inside each ground-truth box and predicts the object's class. Both are evaluated by macro-AP on the common validation split. Features are standardised, and the L2 penalty is selected by three-fold cross-validation on the corresponding training partition.

The GAP probe measures image-level category information, whereas the region probe tests whether tokens at an object's location identify that object, more closely matching how SGDL exposes the token grid to the detector.

Statistical procedure. All correlations are computed across the five encoder means. For each reported p-value, we enumerate all $5! = 120$ assignments of detector AP to the five encoders and report the fraction whose correlation is at least as extreme as the observed value in absolute magnitude. These are therefore two-sided exact permutation p-values. With five encoders, even a perfect rank correlation $\rho = \pm 1$ has a minimum attainable value of $p = 2/120 \approx 0.017$. The p-values are not adjusted for multiple comparisons.

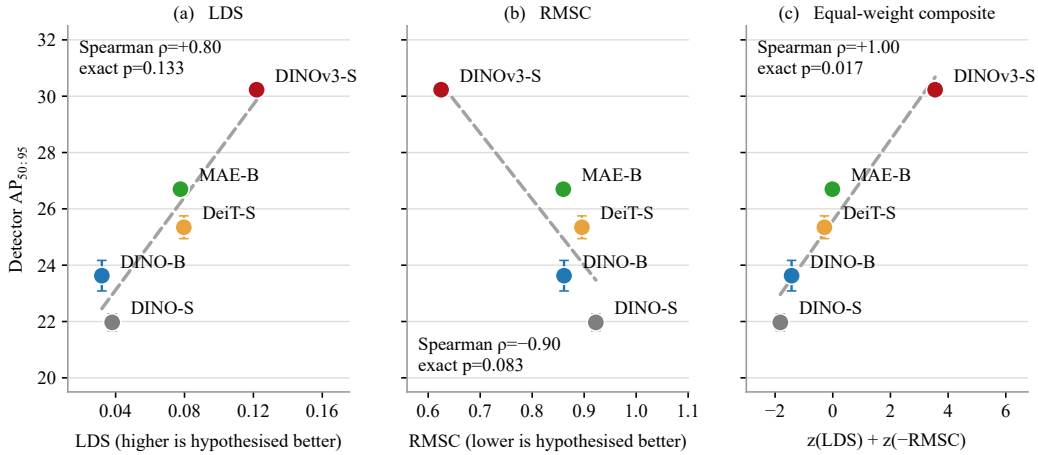

Figure 6: **Structural measures of five frozen encoders against detector AP. (a)** LDS, **(b)** RMSC, and **(c)** their post-hoc equal-weight composite $z(\mathrm{LDS}) + z(-\mathrm{RMSC})$, plotted against the detector's test $\mathrm{AP}_{50:95}$ with each encoder. Vertical bars show ± 1 s.d. over three disjoint partitions; LDS and RMSC are fixed per encoder. Each panel reports Spearman ρ and its exact permutation p-value; for the post-hoc composite in (c), this value is descriptive rather than confirmatory.

Structural statistics and detector AP. Both structural statistics track detector AP closely (Figure 6a,b). Higher LDS accompanies higher AP (Spearman $\rho = +0.80$, Pearson $r = +0.95$), while lower RMSC accompanies higher AP ($\rho = -0.90$, $r = -0.89$). Neither statistic reproduces the complete encoder ordering. LDS swaps MAE-B with DeiT-S and DINO-B with DINO-S, while RMSC swaps DeiT-S with DINO-B (Figure 7c).

The two statistics are strongly anti-correlated in value ($r = -0.80$) but only moderately in rank ($\rho = -0.50$), so their ranking errors occur on different encoder pairs. Having observed that detector AP favours encoders with high LDS and low RMSC, we derive a composite that rewards both equally: the sum of the standardised LDS and the standardised negative RMSC,

$$z(\mathrm{LDS}) + z(-\mathrm{RMSC}),$$

where $z(\cdot)$ standardises a statistic to zero mean and unit variance across the five encoders. This composite is our own construction, not a measure from Singh et al. (2026). It reproduces the detector ordering exactly ($\rho = +1.00$, $p = 0.017$; $r = +0.97$) (Figure 6c). Likewise, among the seven encoder pairs for which one encoder has both higher LDS and lower RMSC, that encoder also has higher detector AP in every case. Because we derived the composite after inspecting these five encoders, its p-value is descriptive rather than confirmatory, and the composite is a candidate signal to be evaluated on additional encoders.

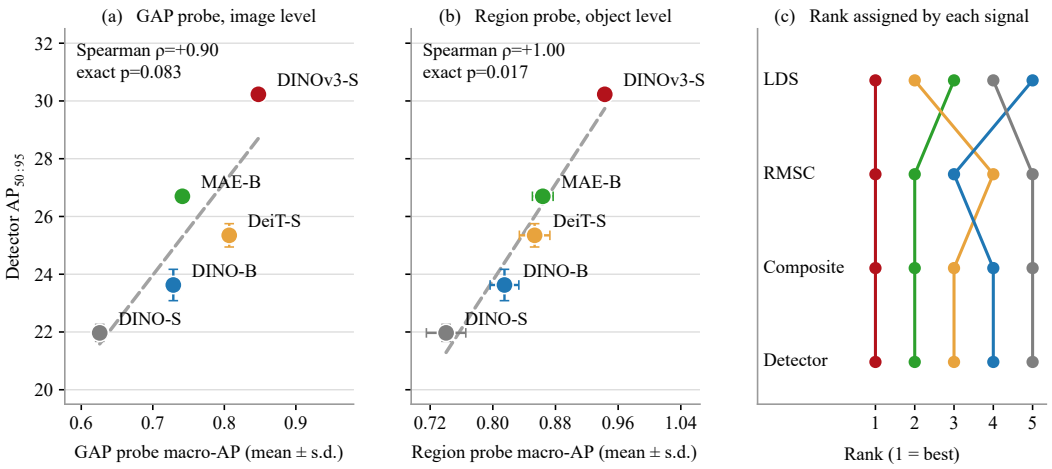

Figure 7: **Linear probes on frozen patch tokens and the ranking assigned by each signal. (a)** GAP-probe and **(b)** region-probe macro-AP on the validation split against detector test $AP_{50:95}$; bars show ± 1 s.d. over three disjoint partitions. **(c)** Encoder ranks assigned by LDS, RMSC, their composite, and detector AP.

Linear probes and detector AP. The region probe also reproduces the detector ordering exactly ($\rho = +1.00$, $p = 0.017$), whereas the GAP probe agrees less closely ($\rho = +0.90$, $p = 0.083$; Figure 7a,b). The region probe likely agrees better because it reads the tokens at an object's location, as the detector does, whereas the GAP probe collapses the token grid into a single image-level vector. Unlike either linear probe, however, LDS and RMSC require no labels.

Prediction beyond the five encoders. A linear regression of detector AP on LDS and RMSC fits the observed encoders closely ($R^2 = 0.95$). Under leave-one-encoder-out prediction, however, $R^2 = -0.58$, worse than predicting the mean AP. The statistics therefore describe the ordering observed in these five encoders but do not yet predict the performance of an unseen encoder.

Interpretation. Across the five encoders, higher detector AP is associated with a combination of strong local organisation (high LDS) and low overall token dispersion (low RMSC). We hypothesise that useful encoders preserve object-level spatial grouping while retaining a substantial shared component across the image, both of which can be passed through a single deep injection.

The RMSC direction is opposite to that reported by Singh et al. (2026), who associate higher RMSC with better generation. Their frozen representation acts as an alignment target for a diffusion transformer, whereas SGDL directly fuses the representation into a convolutional detector. The preferred direction therefore need not transfer between settings. Determining why the relationship reverses requires a larger and more diverse encoder study.

D COMPUTATION COST

Table 9 compares each host in the main results with and without the frozen branch. The branch has two parts: the 21.6 M-parameter frozen DINOv3 encoder and a trainable interface that projects and fuses its features. The interface adds 0.72 M trainable parameters to YOLOv8s and to Faster R-CNN ResNet-18, and 9.18 M to the wider Faster R-CNN ResNet-50 used on COCO (Equation 15). Memory grows by at most 0.24 GiB on every host, because the frozen encoder needs no gradients or optimiser states.

Table 9: **Computation cost of the baseline and hybrid detectors.** Params (M) gives total and trainable parameters in millions. Latency is the mean over 200 timed iterations after 50 warm-up iterations at batch 1 in FP32 on an NVIDIA GeForce RTX 4090, and FPS is its reciprocal. Infer and Train GiB are peak `max_memory_allocated` during inference and during training at batch 4 with mixed precision.

Configuration	Params (M) total / trainable	GMACs	Latency (ms)	FPS	Infer GiB	Train GiB
Stratified regime, 640 px						
YOLOv8s baseline	11.1 / 11.1	14.24	9.75	102.6	0.213	1.111
YOLOv8s oracle	33.5 / 11.9	73.05	31.98	31.3	0.309	1.343
FRCNN-R18 baseline	28.4 / 28.4	69.89	17.76	56.3	0.395	6.424
FRCNN-R18 oracle	50.7 / 29.1	128.71	39.03	25.6	0.513	6.660
COCO regime, 512 px						
FRCNN-R50 baseline	41.7 / 41.7	63.43	18.17	55.0	0.440	6.845
FRCNN-R50 oracle	72.5 / 50.9	98.13	34.77	28.8	0.557	7.080

Compute is the main cost. At a 640 px encoder input, the branch adds 58.8 GMACs (billions of multiply–accumulate operations per image) to both stratified hosts, since the same encoder runs at the same resolution. This raises the compute of YOLOv8s to $5.1\times$ its baseline and that of Faster R-CNN ResNet-18 to $1.8\times$; the absolute cost is set by the encoder, so the relative cost is largest for the lightest host. On COCO, the encoder receives 512 px inputs and raises the compute of Faster R-CNN ResNet-50 to $1.5\times$ its baseline. This host has 41.7 M parameters, about twice the 21.6 M of the frozen encoder, and the prior still improves it (Table 3); the capacity ablation shows gains of similar size for ResNet-50 and for the larger WRN-50-2 on PASCAL VOC (Table 4b). The benefit is therefore not limited to hosts smaller than the encoder.

Speed follows compute. We measure latency in native PyTorch (Paszke et al., 2019), without half precision, quantisation or a compiled runtime such as TensorRT. Under these conditions, the hybrids run at 30 to 52% of their baselines' frame rates, the lowest being YOLOv8s at 31.3 against 102.6 frames per second. Lowering the encoder's input resolution is the most direct way to reduce this cost, and it does not require changing the detector's own resolution (Table 7a).

E ERROR DECOMPOSITION WITH TIDE

This appendix gives the definitions and protocol behind the TIDE analysis of Section 4.2 and Table 2. Rather than counting mistakes, TIDE (Bolya et al., 2020) measures how much AP_{50} would be recovered if one type of error were corrected. It reports error at two levels: a fine-grained decomposition into six main error types and a coarser split into false positives (FP) and false negatives (FN).

Fine-grained error decomposition. TIDE first matches detections to ground truth as in standard AP evaluation, and then groups the resulting false positives and false negatives into six error types. For a false positive, let $\mathrm{IoU}_{\max}$ be its largest overlap with a ground-truth box, and let $t_f = 0.5$ and $t_b = 0.1$ be the foreground and background thresholds.

- **Classification (Cls):** $\mathrm{IoU}_{\max} \geq t_f$ with an object of another class; the box is placed correctly but labelled incorrectly.
- **Localisation (Loc):** $t_b \leq \mathrm{IoU}_{\max} \leq t_f$ with an object of the correct class; the label is correct but the box is misplaced.
- **Both:** $t_b \leq \mathrm{IoU}_{\max} \leq t_f$ with an object of another class; the detection is both mislabelled and misplaced.
- **Duplicate (Dupe):** $\mathrm{IoU}_{\max} \geq t_f$ with an object of the correct class that a higher-scoring detection has already matched.
- **Background (Bkg):** $\mathrm{IoU}_{\max} \leq t_b$ for every ground-truth object; the detection covers no annotated object.
- **Missed (Miss):** a ground-truth object that remains undetected after classification and localisation errors are accounted for.

A count of errors would treat a high-scoring false positive like a low-scoring one, although the former can lower AP more. TIDE therefore measures each error type e by its effect on AP. For each type, it applies an error-specific *oracle* to the detector's original predictions and recomputes AP, for example by relabelling classification errors or suppressing background detections. This TIDE oracle is unrelated to the Oracle of our figures and tables, which denotes the SGDL-augmented detector; TIDE oracles are applied to the predictions of the baseline and the hybrid alike. The contribution of error type e is

$$\Delta\mathrm{AP}_e = \mathrm{AP}_e^{\mathrm{oracle}} - \mathrm{AP}.$$

Every oracle starts from the detector's original predictions, not from a state in which other error types are already corrected, so the contributions of different types can be compared directly. Intuitively, $\Delta\mathrm{AP}_{\mathrm{Cls}} = 2.3$ means that applying the classification oracle alone would recover 2.3 AP_{50} points. Because each oracle is applied separately, the contributions are not additive and do not sum to the total AP lost.

Aggregate FP/FN decomposition. TIDE also regroups the same errors into two aggregate types. The FP oracle suppresses all false-positive detections, and the FN oracle removes false negatives from the recall calculation by setting the number of ground-truth objects to the number of true positives. FP therefore measures the AP lost to erroneous detections, and FN the AP lost to ground-truth objects not recovered as true positives. Because FP and FN regroup several of the six main types, we read them as a separate pair and do not combine them with the fine-grained decomposition.

Evaluation protocol. We run TIDE on the held-out test split of each dataset (4,952 PASCAL VOC and 765 Palm Oil images) at IoU 0.50, with a score threshold of 0.001 and at most 100 detections per image. TIDE computes AP with its own COCO-style evaluator; for every detector in Table 2, we verified that it reproduces the pycocotools AP_{50} of the same prediction files. The AP_{50} values in Table 2 therefore come from this common evaluator for both hosts, whereas Table 1 evaluates each host with its own framework's protocol (Appendix I). Both and Dupe are omitted from Table 2 because they remain at or below 1.5 AP_{50} in every evaluated setting. Appendix M visualises the C5 representations of the full-data PASCAL VOC Faster R-CNN pair analysed here.

F INTERPRETIVE RISK DECOMPOSITION

Our experiments vary the host initialisation, the number of labelled training images, and the target domain. The resulting scaling curves raise two questions, namely why the frozen prior helps and whether that advantage remains as target supervision grows. We interpret these patterns with the classical approximation–estimation–optimisation decomposition of excess risk (Bottou & Bousquet, 2007). The framework applies to a supervised prediction problem defined by a data distribution, a model family, and a loss. We use it to state the qualitative pattern expected from two possible sources of the SGDL gain and then compare those patterns with the stratified experiments of Section 4.2.

F.1 EXCESS-RISK DECOMPOSITION

Let X denote an image and Y its set of labelled object instances. Let $\mathcal{F}$ be a family of detectors, for example all weight settings of one fixed detector architecture, and define the population risk

$$R(f) = \mathbb{E}_{(X,Y)\sim P}[\ell(f(X), Y)]\,, \tag{5}$$

where P is the target-data distribution and ℓ is the loss that defines the prediction problem. For a training set $S_n = \{(X_i, Y_i)\}_{i=1}^n \sim P^n$, define the empirical risk

$$\widehat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(X_i), Y_i). \tag{6}$$

Let $f^\star = \arg\min_f R(f)$ be the population-optimal predictor over all predictors, $f^\star_{\mathcal{F}} = \arg\min_{f\in\mathcal{F}} R(f)$ the best predictor within $\mathcal{F}$, $f_n \in \arg\min_{f\in\mathcal{F}} \widehat{R}_n(f)$ the empirical-risk minimiser for S_n, and $\tilde{f}_n$ the predictor returned by training. The excess risk then decomposes as

$$\underbrace{\mathbb{E}_{S_n}\left[R(\tilde{f}_n) - R(f^\star)\right]}_{\mathcal{E}} = \underbrace{R(f^\star_{\mathcal{F}}) - R(f^\star)}_{\mathcal{E}_{\text{app}}} + \underbrace{\mathbb{E}_{S_n}[R(f_n) - R(f^\star_{\mathcal{F}})]}_{\mathcal{E}_{\text{est}}} + \underbrace{\mathbb{E}_{S_n}\left[R(\tilde{f}_n) - R(f_n)\right]}_{\mathcal{E}_{\text{opt}}}. \tag{7}$$

The reference level $R(f^\star)$ is the minimum population risk achievable for the task under the chosen loss. It is unknown for our detection problems and is not estimated here. The approximation error $\mathcal{E}_{\text{app}}$ is the additional risk caused by restricting prediction to the family $\mathcal{F}$. Once $\mathcal{F}$ is fixed, this term does not depend on the number of labelled target examples. The estimation error $\mathcal{E}_{\text{est}}$ is the additional cost of learning that family from a finite sample rather than from the population distribution; under the usual learning assumptions it decreases as the amount of labelled data increases. The optimisation error $\mathcal{E}_{\text{opt}}$ is the remaining population-risk gap between the model returned by training and the empirical-risk minimiser for that same sample.

For example, suppose the minimum task risk is 3 arbitrary units, the best baseline CNN has risk 7, and the best corresponding SGDL hybrid has risk 5. With a small training sample, suppose the ideal finite-sample solutions have risks 15 and 9, respectively, and optimisation is negligible. Table 10 gives the resulting decomposition.

Table 10: **Illustrative excess-risk decomposition.** Arbitrary units only; these are not experimental measurements.

Quantity	Baseline CNN	SGDL hybrid
Minimum task risk $R(f^\star)$	3	3
Best-in-family risk $R(f^\star_{\mathcal{F}})$	7	5
Expected finite-sample risk $\mathbb{E}_{S_n}[R(f_n)]$	15	9
Approximation error $\mathcal{E}_{\text{app}}$	$7-3=4$	$5-3=2$
Estimation error $\mathcal{E}_{\text{est}}$	$15-7=8$	$9-5=4$
Optimisation error $\mathcal{E}_{\text{opt}}$	≈ 0	≈ 0

Here the hybrid's advantage of 6 units comes from both terms: 2 from a lower best-in-family risk and 4 from a smaller estimation error. A gap of the same size could also arise from either term alone, so its size at a single small budget does not reveal its source.

The quantities in Equation 7 are defined in terms of population risk. Our experiments report AP, which is a dataset-level detection metric rather than the per-example loss in Equation 5. We therefore do not identify AP with population risk and do not numerically estimate the terms in Equation 7. The decomposition is used only to predict how a Baseline–Oracle difference should change as n grows, and those predictions are compared with the observed AP curves.

F.2 Optimisation at the experimental endpoint

The stratified study is a *small-scale learning problem* in the terminology of Bottou & Bousquet (2007): the relevant limitation is the amount of labelled target data rather than the available optimisation time. All stratified models were trained until convergence under the protocol of Appendix I: 288 of the 352 runs stopped early once validation AP plateaued, and the remaining 64 gained at most 0.77 AP over their final 10% of epochs. Within each host, matched Baseline and Oracle conditions use the same data partition, initialisation condition, optimiser, schedule, augmentation, stopping rule, and evaluation procedure.

At the experimental endpoint, we therefore treat residual optimisation error as negligible for the scaling analysis,

$$\mathcal{E}_{\text{opt}} \approx 0, \qquad \mathcal{E} \approx \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}}. \tag{8}$$

Because matched pairs share one protocol, any remaining optimisation error should affect both conditions alike. This does not require the Baseline and Oracle to follow the same training trajectory. Faster convergence of the hybrid is itself an empirical result; the interpretation below concerns their converged endpoints. The eroded YOLOv8s baselines of Section 5 are analysed separately and are not used to distinguish H1 from H2.

F.3 How the decomposition maps to SGDL

For a matched Baseline–Oracle pair, let $\mathcal{F}_B$ denote the host-detector family and $\mathcal{F}_H$ the corresponding SGDL hybrid family. The frozen ViT changes the representation available to the trainable detector, but its encoder parameters are not fitted to the target dataset. Only the host and the coupling layers are updated.

SGDL may therefore reduce either term of Equation 7, and the two effects can occur together.

Approximation error. The fusion block receives both the host's C5 map and the aligned encoder features, so the hybrid can represent detectors whose deepest features combine local evidence with image-wide context. This changes the family of functions available to the trainable detector and may permit a lower best-in-family risk,

$$R(f^\star_{\mathcal{F}_H}) < R(f^\star_{\mathcal{F}_B}),$$

which corresponds to a smaller approximation term. A convolutional host accumulates context through successive local operations, and its effective receptive field can occupy only a fraction of its theoretical receptive field (Luo et al., 2016). The ViT branch instead supplies patch tokens after a global self-attention stack, so each token can contain information aggregated from across the image before fusion. An appearance that is ambiguous locally may therefore become easier to resolve when broader scene context is available.

Estimation error. Training selects the detector that best fits the n labelled images, but the detector is judged on unseen images. The training error $\widehat{R}_n(f)$ is therefore only a noisy estimate of the population risk $R(f)$, and a larger family contains more candidate detectors that fit the training images well by chance while generalising poorly. For a family parametrised linearly by $d_{\mathcal{F}}$ parameters, with bounded inputs, outputs and parameters and a bounded Lipschitz loss, Bottou & Bousquet (2007) make this precise:

$$\mathcal{E}_{\text{est}} \;\le\; 2\,\mathbb{E}_{S_n}\left[\sup_{f\in\mathcal{F}}\left|R(f) - \widehat{R}_n(f)\right|\right] \;\le\; c\sqrt{\frac{d_{\mathcal{F}}}{n}}, \tag{9}$$

where $\sup_{f\in\mathcal{F}}$ takes the largest gap between training error and population risk over the whole family and c is a constant. Intuitively, the estimation error grows with the number of parameters that must be fitted to the target data and shrinks as the number of labelled examples grows. Deep detectors do

not satisfy these assumptions, and parameter count measures their complexity only roughly, so we use Equation 9 only for its direction.

In this bound, $d_{\mathcal{F}}$ counts the parameters fitted to the target data, not all the parameters in the forward pass. The frozen DINOv3 ViT-S/16 encoder contributes 21.6 M parameters that are never fitted to the target labels; it changes the features the trainable detector receives but does not enter $d_{\mathcal{F}}$. SGDL adds only 0.72 M trainable coupling parameters to the stratified hosts (Table 9), so $d_{\mathcal{F}}$ grows by 0.72 M rather than by the 22.3 M added to the forward pass. The frozen parameters still add inference time and memory (Appendix D), but no comparable estimation burden. Total parameter count is therefore not an appropriate proxy for the finite-sample burden in this comparison. The new representation can still change the effective complexity of the family, which parameter count does not capture.

The frozen representation may also reduce the estimation term directly, by making a useful detector easier to learn from few labels. Dubois et al. (2023) define probe-generalisation error as the performance lost by fitting a downstream probe from finite supervised data, and show on ImageNet classification that this term depends strongly on the representation; we use this as an analogue for SGDL.

DINO provides a more direct intuition for why such an analogue is plausible. Frozen DINO features support strong k-NN and linear evaluation on ImageNet, including 78.3% top-1 accuracy with k-NN for a small ViT and 80.1% with a linear classifier for ViT-Base (Caron et al., 2021). These results show that the frozen representation already places semantic classes in a feature space that can be separated by very simple downstream classifiers. DINOv2 and DINOv3 extend this goal to broad frozen-feature transfer across image- and pixel-level vision tasks (Oquab et al., 2024; Siméoni et al., 2026).

Linear separability is related to, but distinct from, the statistical *margin* of classification theory. In binary classification, the margin condition describes how much probability mass lies near an ambiguous decision boundary, where $P(Y = 1 \mid X)$ is close to $1/2$. When little mass lies near that boundary, fewer examples are difficult to classify and faster finite-sample rates can be obtained under suitable assumptions (Tsybakov, 2004; Audibert & Tsybakov, 2007). A frozen feature space in which target-relevant categories are already well separated can be understood in the same qualitative sense, since the trainable downstream model may have less representational organisation to learn from the target labels. The guarantees are for classification and serve here only as intuition.

F.4 Two routes and their scaling signatures

Figure 8 sketches the scaling signature that each route predicts.

H1: Persistent gap. If

$$R(f^{\star}_{\mathcal{F}_B}) > R(f^{\star}_{\mathcal{F}_H}), \tag{10}$$

then the hybrid has a lower best-in-family risk. Estimation error may still be large for both models when n is small, but as finite-sample effects diminish a residual difference can remain because the two model families have different floors.

PASCAL VOC follows this qualitative signature. A substantial positive gap remains at the largest evaluated budget, which is consistent with H1, although the full VOC training set is not an infinite-data limit. The gap at smaller budgets is less informative, because approximation and estimation effects are most strongly mixed when labels are scarce, and its size varies across hosts. The persistent endpoint is therefore the feature that distinguishes this pattern from a purely finite-sample explanation. The capacity ablation of Section 4.4 is consistent with this reading. A wider convolutional host does not close the gap (Table 4b), which suggests that the prior contributes information outside that family's reach rather than additional capacity.

H2: Contracting gap. If the baseline and hybrid have similar best-in-family risks,

$$R(f^{\star}_{\mathcal{F}_B}) \approx R(f^{\star}_{\mathcal{F}_H}),$$

but the frozen representation reduces finite-sample difficulty, then the estimation errors of the hybrid (H) and the baseline (B) at budget n can satisfy

$$\mathcal{E}^{H}_{\mathrm{est}}(n) < \mathcal{E}^{B}_{\mathrm{est}}(n)$$

by a wide margin at small n, with the difference contracting as more labels are supplied. Both H1 and H2 can therefore produce a large low-data gain; the two interpretations become distinguishable only toward the largest available budgets.

Palm Oil follows this contracting pattern. The Oracle gains are large in the low-data regime but shrink strongly with increasing supervision. At full data, YOLOv8s retains only a small positive gain, which is consistent with the frozen prior becoming largely redundant once the host has sufficient target supervision. The pretrained Faster R-CNN Oracle is 2.1 AP below its baseline (Table 1). A negative endpoint is also compatible with redundancy together with interference between the host representation and the frozen features; Section 4.2 examines the error modes behind it.

COCO is evaluated at only one training budget. Its positive SGDL result is compatible with either route, but a single point cannot determine whether the gap would persist or contract as n increases. We therefore do not use COCO to distinguish H1 from H2.

F.5 Limits of the interpretation

The framework is an interpretation of the observed scaling curves rather than a numerical decomposition of AP. The population quantities in Equation 7 are not directly observed because P, $f^\star$, and $f^\star_{\mathcal{F}}$ are unknown. Dubois et al. (2023) construct practical estimators for related representation-learning terms in a restricted ImageNet probing setting, but we do not construct corresponding estimators for object detection.

The largest labelled budget is also finite. A gap that persists through the full VOC training set could still narrow with more data, and the Palm Oil gap could change again beyond the available sample. Approximation and estimation effects can also act together, particularly at small n, so H1 and H2 are not mutually exclusive mechanisms. Finally, only PASCAL VOC and Palm Oil are swept across training budgets. Extending the same controlled scaling analysis to additional domains and larger labelled regimes would test how generally the persistent and contracting signatures recur.

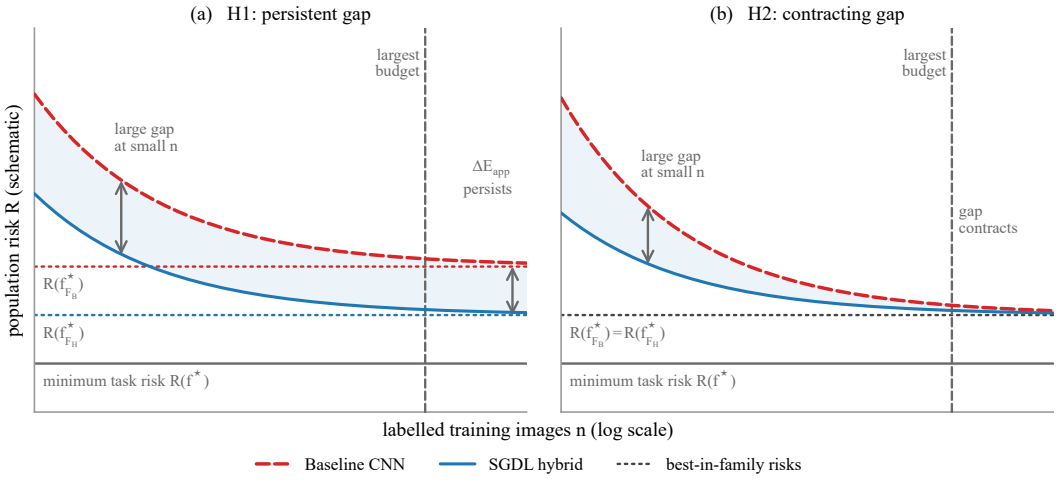

Figure 8: **Schematic of the two scaling signatures.** Illustrative curves, not experimental data. Each panel plots population risk against the number of labelled training images (log scale) for the baseline CNN (dashed) and the SGDL hybrid (solid). The solid horizontal line marks the minimum task risk $R(f^\star)$, dotted lines mark the best-in-family risks $R(f^\star_{\mathcal{F}_B})$ and $R(f^\star_{\mathcal{F}_H})$, and the dashed vertical line marks the largest budget an experiment can test. **(a)** H1: the two families have different best-in-family risks, so as finite-sample and residual optimisation effects diminish, an approximation-error gap $\Delta\mathcal{E}_{\text{app}} = R(f^\star_{\mathcal{F}_B}) - R(f^\star_{\mathcal{F}_H})$ can remain. **(b)** H2: the families share one best-in-family risk, so the hybrid's finite-sample advantage contracts as estimation error diminishes.

G IMPLEMENTATION OF THE SGDL INTERFACE

This appendix specifies the operations shown in Figure 2. SGDL joins the two streams at the last stride-32 stage of the host backbone, before the neck or any context-aggregation block. For a ResNet host this is the output of `layer4`; for YOLOv8s it is the output of the last C2f block, before SPPF. We call this feature map C5 for every host.

Encoder tokens. Let $\mathbf{C} \in \mathbb{R}^{c \times h_C \times w_C}$ be the host's C5 map, with c channels on an $h_C \times w_C$ grid, and let E be a frozen ViT encoder with patch size p and token width d. The encoder receives the detector's input image at $H_V \times W_V$. By default this equals the detector's resolution, so both see the same image at the same size (640×640 in the main experiments); only the resolution ablation of Appendix B resizes the encoder's input. The encoder returns one token per patch, together with prefix tokens such as the class token. We discard the prefix tokens and reshape the patch tokens of the final layer into the grid

$$\mathbf{V} \in \mathbb{R}^{d \times H_V/p \times W_V/p}. \tag{11}$$

Projection and alignment. A first block, ϕ_{proj}, maps $\mathbf{V}$ to m channels with a 1×1 convolution followed by BatchNorm (Ioffe & Szegedy, 2015) and SiLU (Elfwing et al., 2018):

$$\mathbf{P} = \phi_{\text{proj}}(\mathbf{V}) = \text{SiLU}(\text{BN}(\text{Conv}_{1\times1}(\mathbf{V}))) \in \mathbb{R}^{m \times H_V/p \times W_V/p}. \tag{12}$$

We set $m = c$ in all experiments, so the encoder stream has the same width as the host stage and no host-specific design is needed. The encoder features leave the encoder LayerNorm-normalised (Ba et al., 2016), and BatchNorm gives their projection a learned scale and offset before they meet the host's own BatchNorm-normalised features. A parameter-free 2×2 max-pool with stride 2 then halves the grid (Appendix B compares it with learned alternatives). When $p = 16$ and the encoder and the detector receive the same input size, the pooled grid already matches C5, since $2p$ equals the stride of 32. When they do not, the pooled map is resized bilinearly to the size of $\mathbf{C}$:

$$\mathbf{G} = \text{Resize}_{h_C, w_C}(\text{MaxPool}_{2\times2,\, s=2}(\mathbf{P})) \in \mathbb{R}^{m \times h_C \times w_C}, \tag{13}$$

where Resize is the identity when the sizes already agree.

Fusion and training. The aligned map $\mathbf{G}$ is concatenated with $\mathbf{C}$ along the channel axis, and a second 1×1 Conv–BN–SiLU block, ϕ_{fuse}, mixes the two streams and restores the host width:

$$\mathbf{C}' = \phi_{\text{fuse}}([\mathbf{C} \,\|\, \mathbf{G}]) \in \mathbb{R}^{c \times h_C \times w_C}. \tag{14}$$

The fused map $\mathbf{C}'$ replaces $\mathbf{C}$ in the detector's forward pass, so the neck, heads and loss receive a feature map of the shape they expect. During training, the host and both coupling blocks are updated by the host's own detection loss; the encoder stays frozen and supplies features during both training and inference. No additional loss or gating is used. Appendix M compares $\mathbf{C}'$ with $\mathbf{C}$ and with the encoder tokens.

Trainable parameter count. Both 1×1 convolutions have no bias, and each BatchNorm layer has a learned scale and offset per output channel. The interface therefore adds

$$|\Theta| = \underbrace{m(d+2)}_{\phi_{\text{proj}}} + \underbrace{c(c+m+2)}_{\phi_{\text{fuse}}} \tag{15}$$

trainable parameters. With $m = c$ this becomes $cd + 2c^2 + 4c$: 722,944 for $c = 512$ (ResNet-18 and YOLOv8s) and 9,183,232 for $c = 2{,}048$ (ResNet-50). These counts exclude the frozen encoder weights. Most of the parameters lie in ϕ_{fuse}, whose input carries both the c host channels and the m projected channels.

Other injection sites. The injection-site ablation (Table 4a) also attaches the interface to the ResNet stages whose outputs we call C3 (`layer2`, stride 8) and C4 (`layer3`, stride 16), which have 128 and 256 channels on ResNet-18. Every site uses the same projection and fusion blocks, with $m = c$ set to the width of that stage, and the sites differ only in how the stride-16 token grid is aligned with the stage. At C5 the grid is halved by the 2×2 max-pool described above. At C4 it

already matches the stage, so the alignment is the identity. At C3 it is upsampled by a factor of two with bilinear interpolation and then passed through a learned 3×3 Conv–BN–SiLU block, which refines the interpolated map and adds $9c^2 + 2c$ parameters. For a 640×640 input, the aligned maps are therefore $128 \times 80 \times 80$, $256 \times 40 \times 40$ and $512 \times 20 \times 20$. The 3×3 block accounts for most of the parameters added at C3, which is why C3 and C4 add similar counts (0.23 M each) despite their different widths. When several sites are used, the encoder runs once per image and every site reads the same token grid. Each fused map replaces the output of its stage, so it feeds both the next backbone stage and the corresponding FPN level.

H DATASETS, REGIMES AND PARTITIONS

Table 11 lists the three datasets, the role each plays in the study and the training budgets drawn from them. PASCAL VOC and Palm Oil are used for the stratified data-regime study; COCO is trained only at full scale.

Table 11: **Datasets and training budgets.** *Held-out split* is the split on which results are reported. Four disjoint partitions are drawn at each budget $n \leq 700$, three at $n = 1{,}000$ and one at full data.

Dataset	**Classes**	**Role in the study**	**Held-out split**	**Budgets** n
PASCAL VOC	20	Natural scenes with varied context (Everingham et al., 2010); main testbed for the data-regime study and the ablations.	4,952 (test)	100, 300, 500, 700, 1,000, full (16,551)
Palm Oil	5	Large public dataset of about 14,500 smartphone images of oil palm fruit bunches in commercial plantations in Indonesia (Suharjito et al., 2025); a homogeneous scene under harsh field conditions, with classes distinguished mainly by colour and texture.	765 (test)	50, 100, 300, 500, 700, 1,000, full (10,221)
COCO	80	Complex everyday scenes with objects in context (Lin et al., 2014); tests the prior at full scale with a larger taxonomy.	5,000 (`val2017`)	full `train2017`

Absolute budgets. Budgets are defined as numbers of training images rather than as percentages of each dataset. This gives both datasets a common set of budget sizes and lets us draw several disjoint partitions at each budget, which captures more of the variation between training samples; in low-data detection this variation can be large (Wang et al., 2020). The amount of supervision at a given budget still differs between datasets, because images contain different numbers of annotated objects. The smallest budget follows from the number of classes. As a heuristic minimum of supervision, we require at least five training images per class, and round the resulting number of images up to the next multiple of 50. Palm Oil, with five classes, thus needs $5 \times 5 = 25$ images, which rounds up to a smallest budget of 50, whereas PASCAL VOC needs $20 \times 5 = 100$, which is already a multiple of 50. This is why Palm Oil has a budget of 50 and PASCAL VOC does not.

Stratified, disjoint partitions. An image in a detection dataset can contain several classes, so a random subset can easily over- or under-represent rare classes, particularly at small budgets. We therefore draw every partition with iterative stratification for multi-label data (Sechidis et al., 2011), treating each image as labelled with the set of classes it contains. The method assigns images class by class, starting with the rarest, so that the fraction of images containing each class is preserved both in the selected subset and in the remaining pool. The partitions at a budget are drawn in sequence from a pool shuffled once with a fixed seed. The first partition is drawn from the whole training pool, the second from the pool without the first, and so on. A final check confirms that no image appears in two partitions. Three partitions of $n = 1{,}000$ therefore contain 3,000 distinct images, not three resamples of one subset.

Class-balance audit. Each partition is compared with the full training pool immediately after it is drawn. Let p_k be the fraction of pool images that contain class k and q_k the same fraction in the partition, each vector normalised to sum to one. We compute the Jensen–Shannon divergence (Lin, 1991)

$$\mathrm{JS}(p, q) = \tfrac{1}{2}\,\mathrm{KL}(p \,\|\, a) + \tfrac{1}{2}\,\mathrm{KL}(q \,\|\, a), \qquad a = \tfrac{1}{2}(p + q), \tag{16}$$

with base-2 logarithms, so that it is 0 for identical class distributions and 1 for distributions with no class in common. A partition whose divergence exceeds 0.10, or in which any single class deviates markedly from its pool fraction, is flagged for inspection. The audit ensures that partitions differ in size and not in class composition, and that they keep the imbalance of the original dataset. For example, a class present in 5% of the pool images should appear in about 5% of the images of every partition.

Validation and test splits. The validation split is a stratified subset of about 600 images, drawn once from each dataset's validation set rather than using the full set, to reduce the cost of validating after every epoch. It is disjoint from every training partition and shared by all budgets, partitions and conditions. It is used only for early stopping and checkpoint selection; all reported results use the held-out test split. COCO is not partitioned. It trains on the full `train2017` split and validates on the full `val2017` split with one run per condition.

Example. At $n = 300$ on PASCAL VOC, four disjoint partitions of 300 images are drawn as above, 1,200 distinct images in total, each keeping the class distribution of the training pool, imbalance included. Every condition (baseline and oracle, each with a randomly initialised and a pretrained host) is trained once on each partition with the same training seed, so its four runs differ only in their data. Each run is evaluated on the test split, and the $n = 300$ cell of Table 1 is the mean over the four partitions; the spread across them is in Table 16. The corresponding Δ is the oracle minus the baseline on each partition, averaged over the four; every Δ in the main text is computed this way.

I FULL TRAINING PROTOCOL

Each host is trained with the standard pipeline of its framework. Table 12 lists the settings of the main experiments. Any option not listed is the framework's default, and each ablation states its own schedule where it is reported. Every run uses the training seed 42, so the reported spread reflects variation between the disjoint data partitions of Appendix H rather than between seeds. Where a framework replaces a supplied setting at run time, we report the value it used, as read from the training logs.

Table 12: **Training protocol.** Stratified runs are those of Table 1, and the COCO run that of Table 3. Faster R-CNN uses AdamW (Loshchilov & Hutter, 2019) with cosine annealing (Loshchilov & Hutter, 2017). * chosen by `optimizer=auto` (Table 13). § effective batch 64.

	Stratified		COCO
Setting	**YOLOv8s**	**FRCNN-R18**	**FRCNN-R50**
Detector input (px)	640×640	640×640	512×512
Encoder input (px)	640×640	640×640	512×512
Epoch budget	300	100	12 ($1\times$ schedule)
Early-stopping patience	30	20	disabled
Batch size	$4^{\S}$	4	4×4 accum. $= 16$
Optimiser	`auto`*	AdamW	AdamW
Initial learning rate	`auto`*	10^{-4}	10^{-4}
Weight decay	5×10^{-4}	10^{-4}	10^{-4}
Schedule	Ultralytics default	cosine to 10^{-6}	cosine to 10^{-6}

Optimiser on YOLOv8s. The YOLOv8s runs keep Ultralytics' default `optimizer=auto`, with which the trainer chooses the optimiser and its initial learning rate from the training-set size, the epoch budget and the number of classes (Table 13). It selects AdamW at the restricted budgets and MuSGD, Ultralytics' hybrid of the Muon and SGD updates (Ultralytics, 2025; Jordan et al., 2024), at full data. These three quantities are the same for both arms of a matched pair, so every Baseline–Oracle comparison uses the same optimiser and learning rate, as the training logs confirm. The batch settings are also left as Ultralytics sets them. With a batch of 4 and a nominal batch of 64, the trainer accumulates gradients over 16 steps, which gives an effective batch of 64.

Table 13: **Optimiser chosen by `optimizer=auto` on YOLOv8s**, as read from the training logs. The choice is the same for all four arms at each dataset and budget.

Dataset	**Budget**	**Classes**	**Optimiser**	**Initial lr**
PASCAL VOC	100 . . . 1,000	20	AdamW	4.17×10^{-4}
PASCAL VOC	full	20	MuSGD	1.00×10^{-2}
Palm Oil	50 . . . 1,000	5	AdamW	1.11×10^{-3}
Palm Oil	full	5	MuSGD	1.00×10^{-2}

Ultralytics configuration. Table 14 lists the Ultralytics 8.4.37 settings of the YOLOv8s runs. All are the stock defaults for `task=detect` except `close_mosaic`, which we set to 0 so that mosaic augmentation (Bochkovskiy et al., 2020) stays on for the whole schedule.

Faster R-CNN augmentation. The torchvision hosts use light augmentation, namely horizontal flips with probability 0.5 and `ColorJitter` (brightness 0.4, contrast 0.4, saturation 0.4, hue 0.1), without mosaic, rotation or scale jitter. We keep each host's augmentation as it is canonically used with that framework rather than equalising the two, because every comparison in the study is made within one host, where both arms share the same pipeline.

Detection thresholds. Per-epoch validation, used only for checkpoint selection and early stopping, applies a confidence threshold of 0.05 on Faster R-CNN (the torchvision default) and 0.001 on

Table 14: **Ultralytics 8.4.37 settings for the YOLOv8s runs.** † changed from the default. ‡ used only for classification tasks. * replaced at run time by `optimizer=auto` (Table 13).

Augmentation	Value
mosaic	1.0
close_mosaic	0†
hsv_h	0.015
hsv_s	0.7
hsv_v	0.4
degrees	0.0
translate	0.1
scale	0.5
shear	0.0
perspective	0.0
flipud	0.0
fliplr	0.5
bgr	0.0
mixup	0.0
cutmix	0.0
copy_paste	0.0
rect	False
multi_scale	0.0
auto_augment	randaugment‡
erasing	0.4‡

(a) Augmentation

Optimisation	Value
optimizer	auto
nbs	64
batch	4
lr0	0.01*
lrf	0.01
momentum	0.937*
weight_decay	0.0005
cos_lr	False
warmup_epochs	3.0
warmup_momentum	0.8
warmup_bias_lr	0.1*
box	7.5
cls	0.5
dfl	1.5
iou (NMS)	0.7
max_det	300
amp	True
deterministic	True

(b) Optimisation and schedule

YOLOv8s (the Ultralytics default). Every reported number is evaluated at a threshold of 0.001, with each host scored by its own framework's evaluation: Faster R-CNN with the COCO protocol at up to 100 detections per image, and YOLOv8s with the Ultralytics validator at up to 300 (Table 14). No test image contains more than 37 annotated objects in PASCAL VOC or 5 in Palm Oil, so both limits exceed the number of objects in any test image.

J STRATIFIED RESULTS AT AP_{50} AND BETWEEN-PARTITION VARIABILITY

Table 15 reports the comparison of Table 1 at AP_{50}, and Table 16 gives the standard deviation behind each mean in Table 1.

Table 15: **Stratified data-regime comparison at AP_{50}.** Test AP_{50} against training-set size for the runs of Table 1. B is baseline and O oracle; $_{\mathrm{rnd}}$ and $_{\mathrm{pre}}$ are random and pretrained host initialisation. Δ rows are oracle $-$ baseline in AP_{50} points. Bold marks the best entry per column within a host block. * the baseline's accuracy erodes after an early peak, so these cells report its best checkpoint (Section 5).

	PASCAL VOC						Palm Oil						
Init.	**100**	**300**	**500**	**700**	**1,000**	**Full**	**50**	**100**	**300**	**500**	**700**	**1,000**	**Full**
Host: YOLOv8s													
B_{rnd}	3.2	13.3	20.6	26.9	34.3	83.1	24.2	35.4	53.2	62.0	66.6	69.3	75.0
O_{rnd}	44.5	64.5	70.1	73.6	76.5	**91.4**	**51.5**	53.9	65.7	**70.8**	71.8	74.0	**77.4**
Δ_{rnd}	+41.3	+51.2	+49.5	+46.8	+42.2	+8.3	+27.3	+18.5	+12.5	+8.8	+5.2	+4.7	+2.3
B_{pre}	49.3*	63.6*	68.8*	70.5*	73.1*	86.0	34.8	47.3	57.6	65.4	68.1	70.5	73.5
O_{pre}	**53.7**	**70.4**	**73.4**	**75.5**	**78.1**	91.3	48.1	**56.7**	**67.2**	70.0	**71.9**	**75.4**	**77.4**
Δ_{pre}	+4.4	+6.7	+4.7	+5.0	+5.1	+5.3	+13.3	+9.4	+9.5	+4.6	+3.8	+4.9	+3.9
Host: Faster R-CNN (ResNet-18-FPN)													
B_{rnd}	1.8	5.3	7.8	10.0	12.6	53.1	15.9	26.8	40.1	42.3	46.8	53.3	62.4
O_{rnd}	42.9	63.3	69.0	71.4	74.6	85.7	47.4	52.8	**65.2**	67.9	**70.3**	69.9	71.8
Δ_{rnd}	+41.1	+57.9	+61.3	+61.3	+62.0	+32.6	+31.5	+25.9	+25.1	+25.6	+23.5	+16.6	+9.4
B_{pre}	14.4	34.6	39.9	45.2	48.7	73.1	32.9	43.7	56.3	64.8	67.1	69.1	**77.7**
O_{pre}	**45.3**	**66.0**	**71.5**	**74.1**	**76.6**	**87.2**	**48.6**	**53.8**	64.0	**68.2**	69.2	**70.4**	72.6
Δ_{pre}	+30.9	+31.4	+31.6	+28.9	+27.9	+14.1	+15.7	+10.0	+7.7	+3.4	+2.0	+1.3	−5.1

Table 16: **Between-partition standard deviation for Table 1.** Test $AP_{50:95}$ points over four partitions at $n \leq 700$ and three at $n = 1{,}000$; full data has a single partition and is omitted. Δ rows give the deviation of the paired per-partition difference.

	PASCAL VOC					Palm Oil					
Arm	**100**	**300**	**500**	**700**	**1,000**	**50**	**100**	**300**	**500**	**700**	**1,000**
Host: YOLOv8s											
B_{rnd}	0.32	0.29	0.93	0.83	0.50	1.19	2.81	2.36	2.81	1.15	1.88
O_{rnd}	2.00	1.83	2.04	0.25	0.36	2.01	2.58	2.31	2.26	1.12	1.98
Δ_{rnd}	2.19	1.89	1.88	0.62	0.32	2.32	3.78	2.76	4.33	1.30	0.78
B_{pre}	1.29	1.19	1.15	1.48	0.81	2.74	2.11	2.21	0.79	0.44	1.66
O_{pre}	1.71	0.27	1.05	0.89	0.44	1.44	2.64	2.59	2.48	1.38	1.66
Δ_{pre}	1.35	1.45	1.23	2.13	1.01	2.93	3.68	2.83	2.76	1.21	3.24
Host: Faster R-CNN (ResNet-18-FPN)											
B_{rnd}	0.21	0.18	0.13	0.24	0.18	0.50	1.15	1.52	1.10	0.99	1.90
O_{rnd}	0.46	0.71	0.81	0.49	0.39	1.53	1.89	2.54	2.37	0.48	2.82
Δ_{rnd}	0.46	0.60	0.92	0.38	0.57	2.01	1.63	2.12	2.03	1.05	4.68
B_{pre}	0.84	1.38	0.68	0.76	0.76	2.17	1.07	2.07	1.42	0.64	0.89
O_{pre}	0.59	0.72	0.59	0.89	0.20	1.90	2.30	2.84	1.81	1.06	0.51
Δ_{pre}	0.93	0.71	1.21	0.54	0.91	2.65	3.27	3.28	0.74	1.05	0.88

K CONVERGENCE CURVES AT THE STRATIFIED BUDGETS

Figure 9 extends Figure 3 to the budgets it omits, with the same partition, smoothing and axis rules. The pattern of the main figure holds at these budgets, where the Oracle arms generally rise faster than the matching baselines on both hosts and datasets.

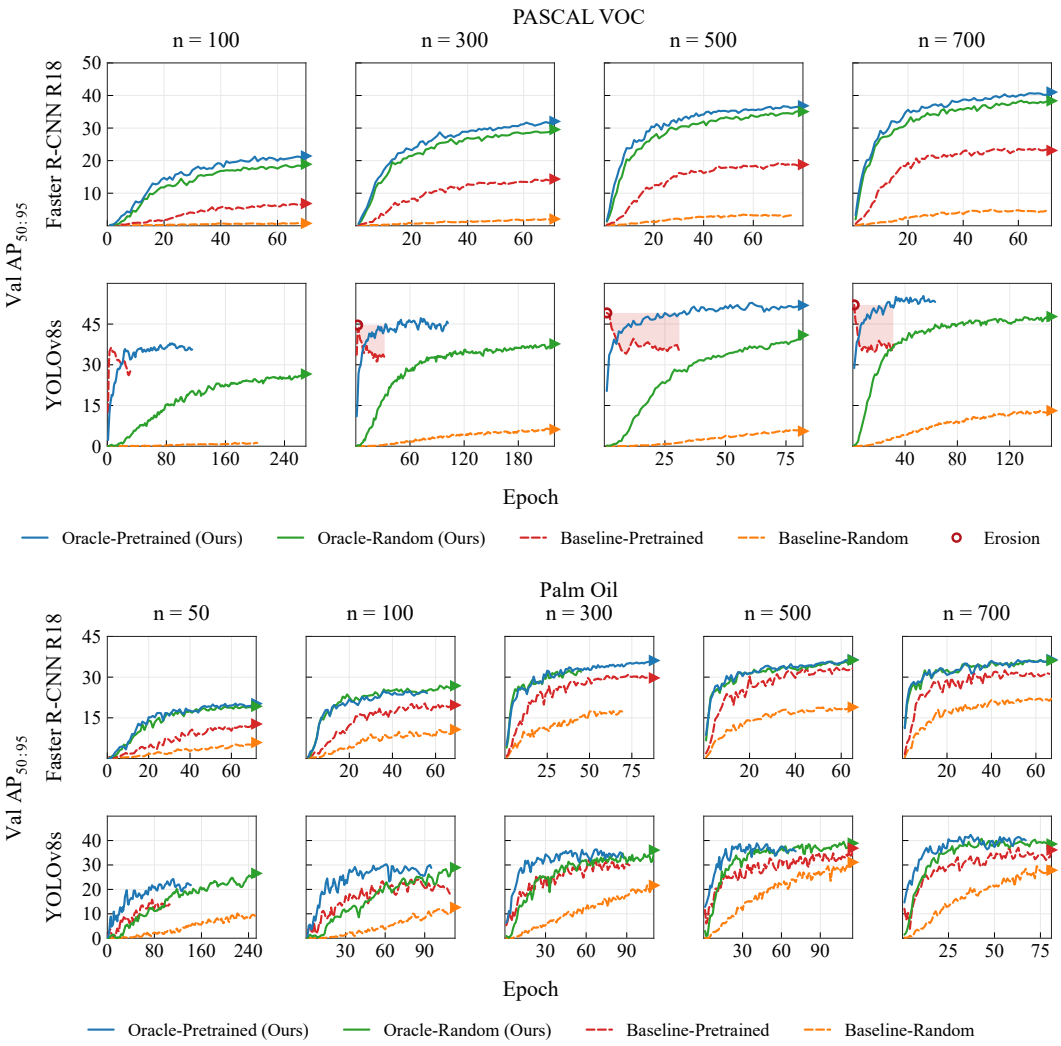

Figure 9: **Convergence curves at the stratified budgets.** Validation $AP_{50:95}$ against training epoch for the four arms of the factorial at the budgets not shown in Figure 3: PASCAL VOC at $n = 100$ to 700 (top) and Palm Oil at $n = 50$ to 700 (bottom). Rows are hosts, and each panel shows one representative partition. Arrows mark runs that continue beyond the plotted range. On PASCAL VOC, the circle marks the pretrained YOLOv8s baseline's early peak and shading the accuracy it loses afterwards; Palm Oil carries no such marks, since its classes lie outside the COCO label space.

L COMPARISON WITH CLOSELY RELATED WORK

Table 17 compares SGDL with the methods of Section 2 that are closest to it, grouped by the role of the pretrained model, and describes the interface through which each reaches the detector.

Table 17: **Closely related detection methods,** grouped by the role of the pretrained model (in parentheses). *Flow*, *Site(s)* and *Fusion / bridge* give the direction, location and mechanism of the interaction; *Host change* is what the method changes in the detector. Tasks are closed-set detection unless stated.

Method (pretrained model)	Flow	Site(s)	Fusion / bridge	Host change	Detector; task
Transformer as the detector backbone, fine-tuned					
ViTDet (MAE ViT) (Li et al., 2022b)	ViT → head	Last feature map	Strided convolutions and deconvolutions form a simple pyramid	ViT replaces the backbone	(Cascade) Mask R-CNN
ViT-Adapter (ViT) (Chen et al., 2023)	CNN ↔ ViT	Four blocks along the ViT	Spatial prior; deformable cross-attention injector and extractor	Adapter added to the ViT	Mask R-CNN and others
ViT-CoMer (ViT) (Xia et al., 2024)	CNN ↔ ViT	Every ViT stage	MRFP pyramid; CTI fusion with multi-scale deformable attention	CNN branch added to the ViT	Mask R-CNN and others
DEIMv2 (DINOv3 ViT) (Huang et al., 2025)	ViT → encoder	Blocks 5, 8, 11 at 1/16 scale	STA: bilinear rescaling, light convolutional branch, Bi-Fusion	DINOv3 with STA as backbone	Real-time DETR
Frozen model as the detector backbone					
Vasconcelos et al. (2022) (ImageNet CNN)	Backbone → head	Backbone features	None; larger trainable neck and cascade heads	Frozen backbone	Faster and Mask R-CNN
F-VLM (CLIP) (Kuo et al., 2023)	Backbone → head	Backbone features	None; detector and VLM scores combined at test time	Frozen backbone	Mask R-CNN; open-vocabulary
F-ViT (CLIP ViT) (Wu et al., 2024)	Backbone → head	Intermediate ViT layers	None; intermediate maps interpolated to several scales	Frozen backbone	Two-stage; open-vocabulary
Frozen model as an auxiliary to a trainable detector					
Frozen-DETR (CLIP, DINOv2) (Fu et al., 2024)	ViT → encoder, decoder	Encoder input; every decoder stage	Patch map as an extra pyramid level; class token as a decoder image query	DETR encoder and decoder inputs extended	DETR family
VMCNet (CLIP ViT) (Gao et al., 2025)	ViT → CNN	ViT layers 1, 5, 7; all CNN scales	Layers fused linearly; cascaded multi-scale deformable attention	Two-branch CNN–ViT backbone	Two-stage; open-vocabulary
DINO-YOLO (DINOv3 ViT-B, ViT-L) (Malaisree et al., 2025)	ViT → input; CNN → ViT → CNN	One to three of P0, P3, P4	P0: three-channel input; P3: frozen DINOv3 blocks, gated fusion	YOLOv12 input and backbone modified	YOLOv12; civil engineering
DSOD (DINOv2 ViT-B) (Cai et al., 2026)	ViT → CNN	C3, C4, C5	Weighted addition with DAAW weights; SAFR regularisation loss	Dual-tower backbone; teacher–student	Deformable DETR; source-free
SGDL (DINOv3 ViT-S/16)	ViT → CNN	Final patch map; C5 only	Projection, pooling, concatenation, 1×1 Conv–BN–SiLU; no attention, gate or extra loss	C5 replaced; neck, heads and loss unchanged	YOLOv8s, Faster R-CNN

SGDL uses the lightest coupling among the auxiliary methods. Its frozen encoder contributes only the final patch map, which is projected, pooled to the C5 grid, concatenated with C5 and fused by one 1×1 Conv–BN–SiLU block. The fused map has the shape of C5 and replaces it, so the host's neck, heads and loss are unchanged, and the interface adds no attention bridge, gate or auxiliary loss. The closest auxiliary methods use richer bridges: Frozen-DETR extends both the encoder and the decoder of a DETR, VMCNet builds a two-branch backbone whose modulation blocks use multi-scale deformable attention, DINO-YOLO inserts DINOv3 modules at up to three sites and passes mid-backbone CNN features through DINOv3 blocks with gated fusion, and DSOD fuses at three stages with adaptive weights and a regularisation loss. Because the SGDL interface is fixed and independent of the detector, the same coupling serves a one-stage and a two-stage host, and our experiments vary host initialisation, supervision, target domain and encoder while the coupling stays constant.

M QUALITATIVE ANALYSIS OF THE FUSED C5 REPRESENTATION

We visualise the representation that SGDL passes to the detector, for the full-data, ImageNet-pretrained Faster R-CNN ResNet-18 pair on PASCAL VOC whose errors Table 2 decomposes. On fifteen randomly drawn test images, we compare the baseline C5 map, the fused C5 map of the hybrid (*SGDL C5*) and the frozen DINOv3 patch tokens (*raw ViT*), using the largest ground-truth box in each image as the query.

Cosine similarity maps. Each map gives the cosine similarity of every position to the feature at the query centre (Figures 10–12). Baseline C5 responds weakly, in a small region around the query point, whereas the raw ViT response is coherent but spreads across much of the scene. The fused map lies between the two, following the object-level shape of the raw ViT response while staying largely within the object. For the motorbike in the last row of Figure 10, it covers both the vehicle and its rider, and for the cat in Figure 11, it spans almost the whole animal.

PCA maps. Figures 13–15 show the first three principal components of each map as RGB. The baseline C5 maps show little spatial structure that corresponds to objects. The fused maps separate object from background with a layout close to that of the raw ViT, at a coarser resolution, for example the person in the first row of Figure 13 and the car interior in Figure 14.

Relation to the error decomposition. The maps show the intended effect of SGDL: the object-level organisation of the frozen encoder reaches C5, so the detector's heads receive features that describe the whole object and not only a local patch around its centre. This is consistent with the TIDE results for the same pair (Section 4.2), in which classification error falls by more than any other error type and false-positive error halves. Intuitively, features that are consistent across an object and distinct from its surroundings give the classifier coherent evidence for the whole region and leave fewer background regions to be mistaken for objects.

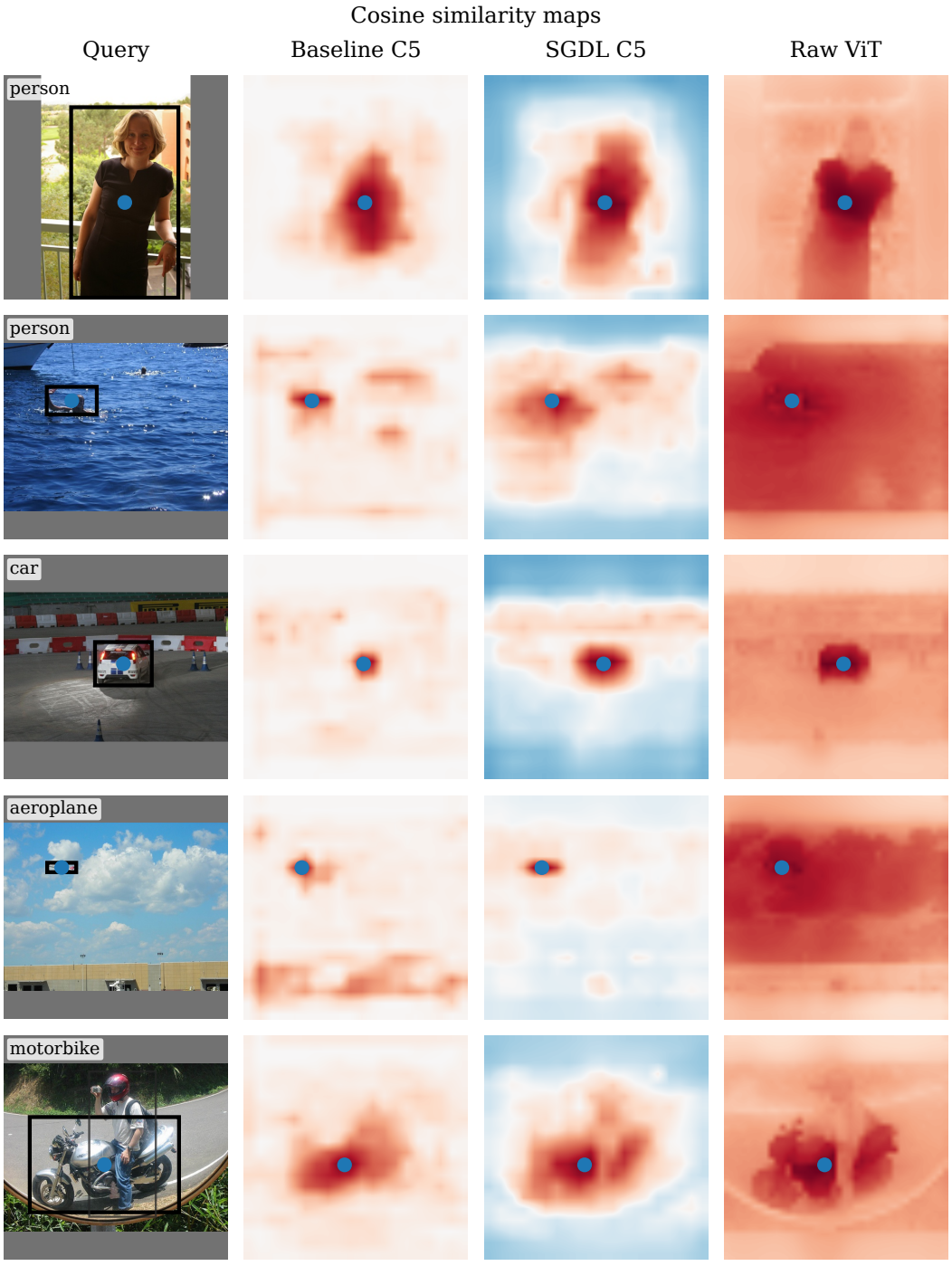

Figure 10: **Cosine similarity maps for five PASCAL VOC test images.** *Query*: annotated boxes, with the thick box as the query, the blue dot its centre and the label its category. Other columns: similarity to the query-centre feature in baseline C5, fused C5 and the frozen DINOv3 tokens (Faster R-CNN ResNet-18, full data, pretrained host). Red, white and blue mark positive, zero and negative similarity on a common $[-1, 1]$ scale; baseline C5 follows a ReLU and is therefore non-negative.

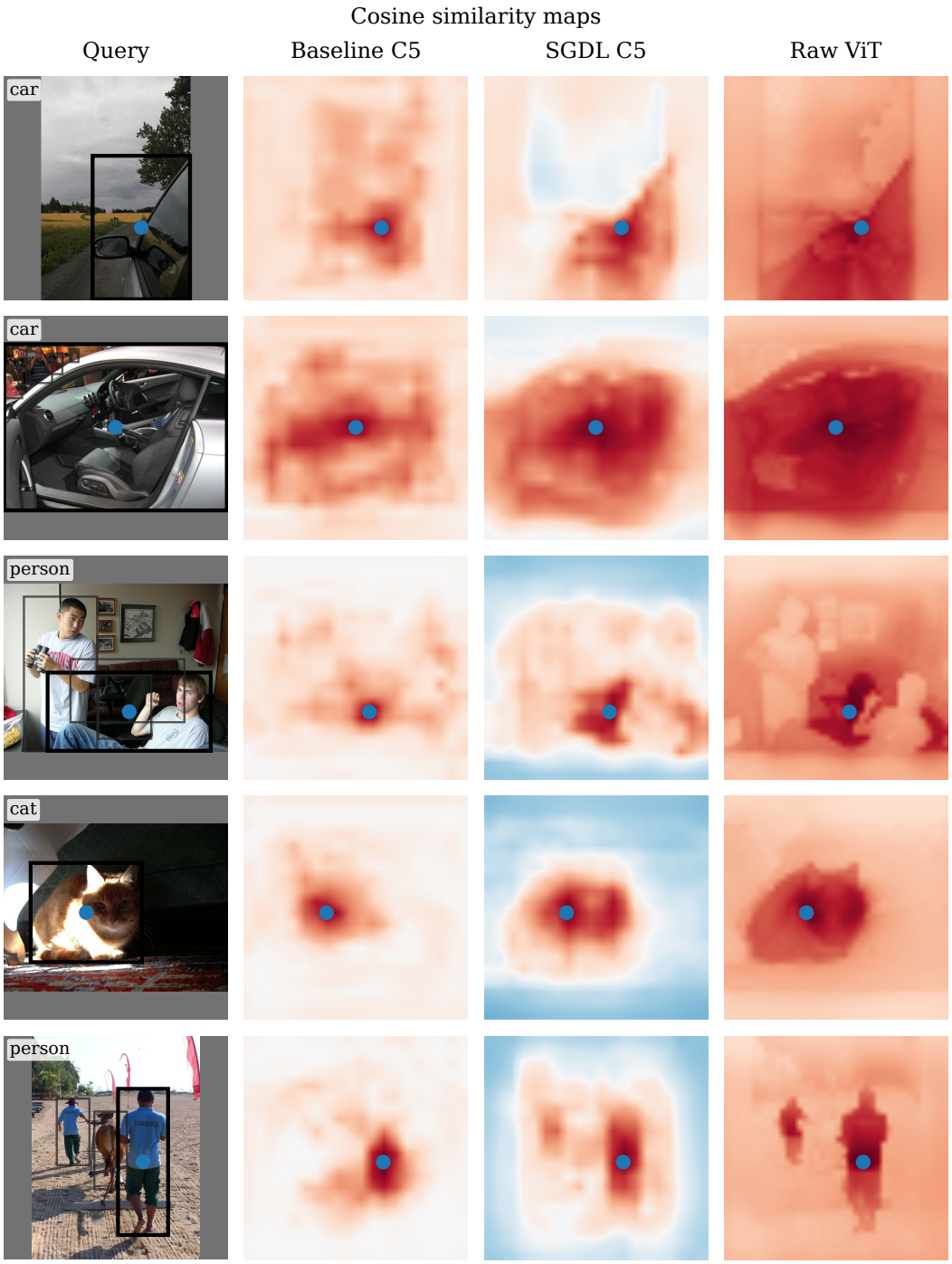

Figure 11: **Cosine similarity maps, continued,** for five further PASCAL VOC test images. Layout and colour scale as in Figure 10.

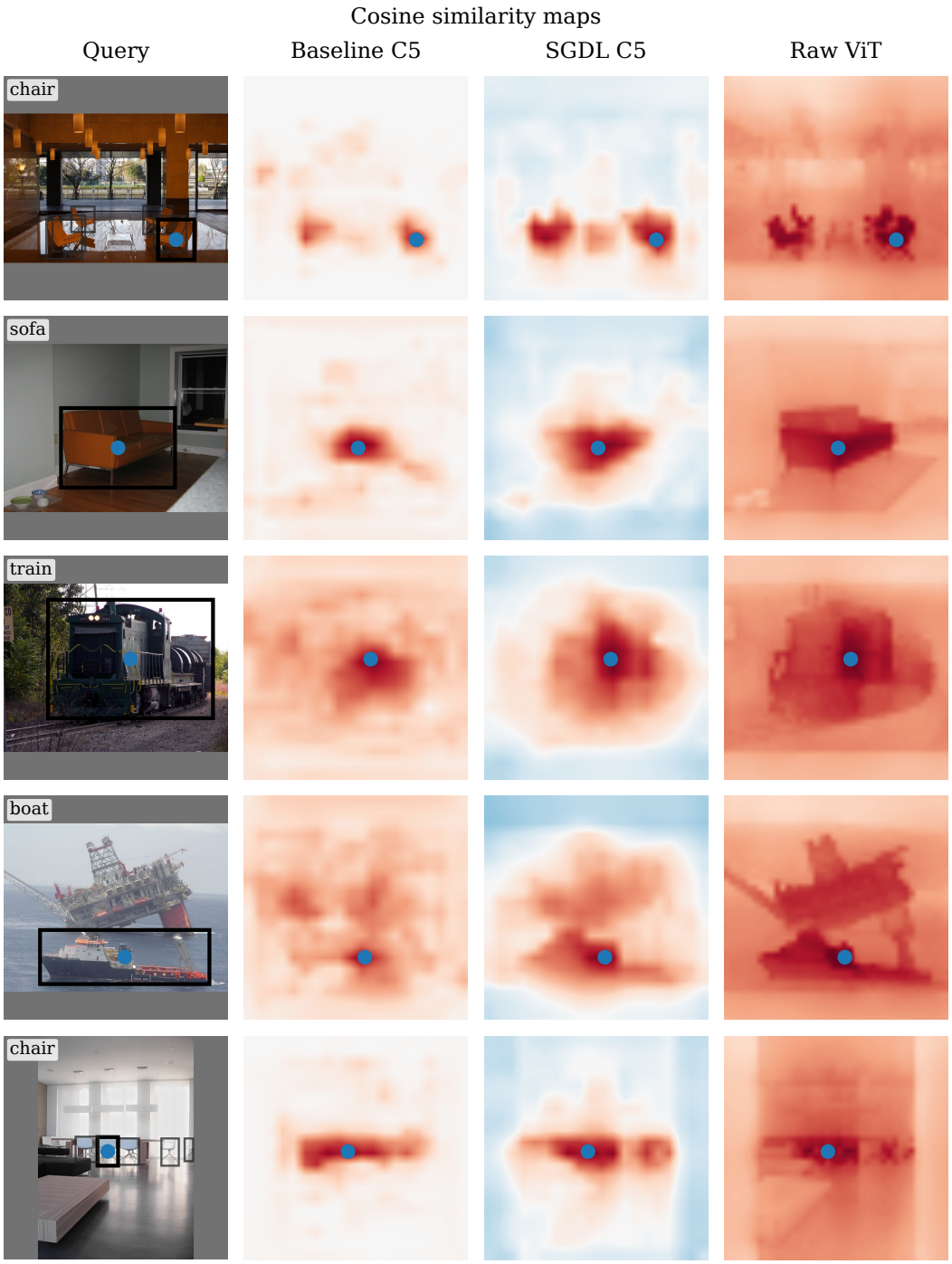

Figure 12: **Cosine similarity maps, continued,** for the last five PASCAL VOC test images. Layout and colour scale as in Figure 10.

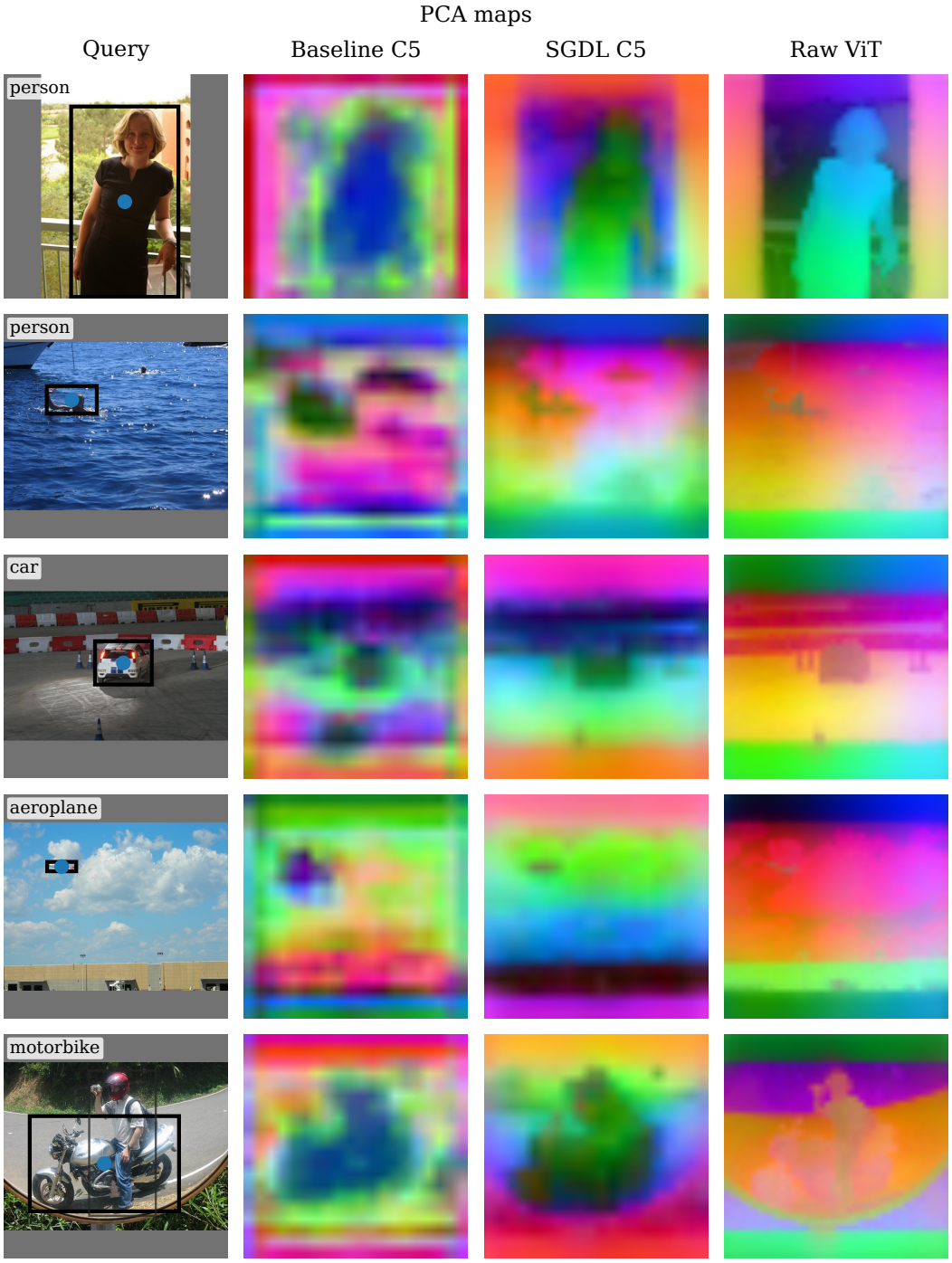

Figure 13: **First three principal components of each representation as RGB,** for the images and columns of Figure 10. PCA is fitted per map on L2-normalised features, with each component scaled to its 1st–99th percentile range, so colours are not comparable across columns.

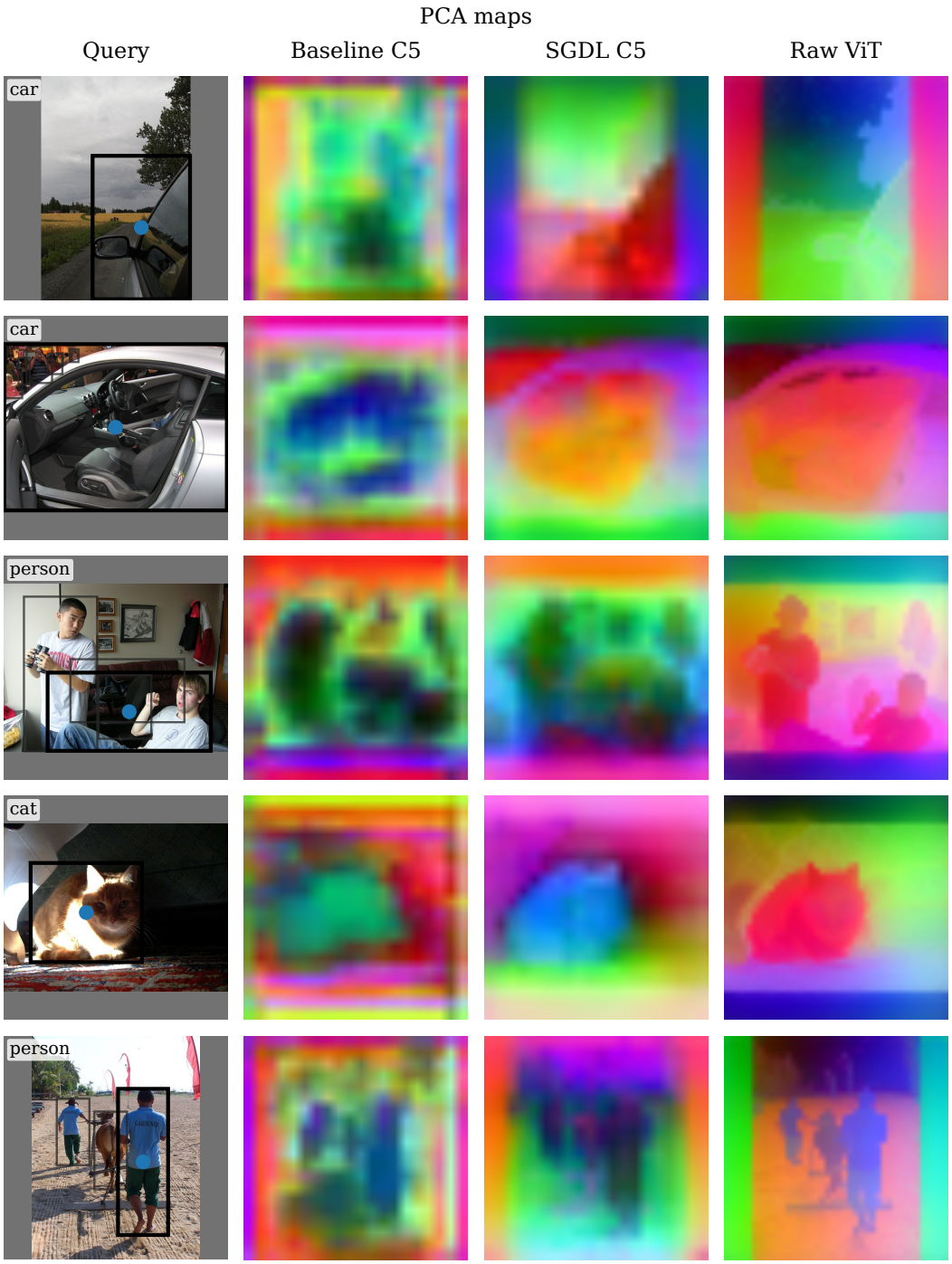

Figure 14: **Principal-component maps, continued,** for the images of Figure 11. Layout and colour mapping as in Figure 13.

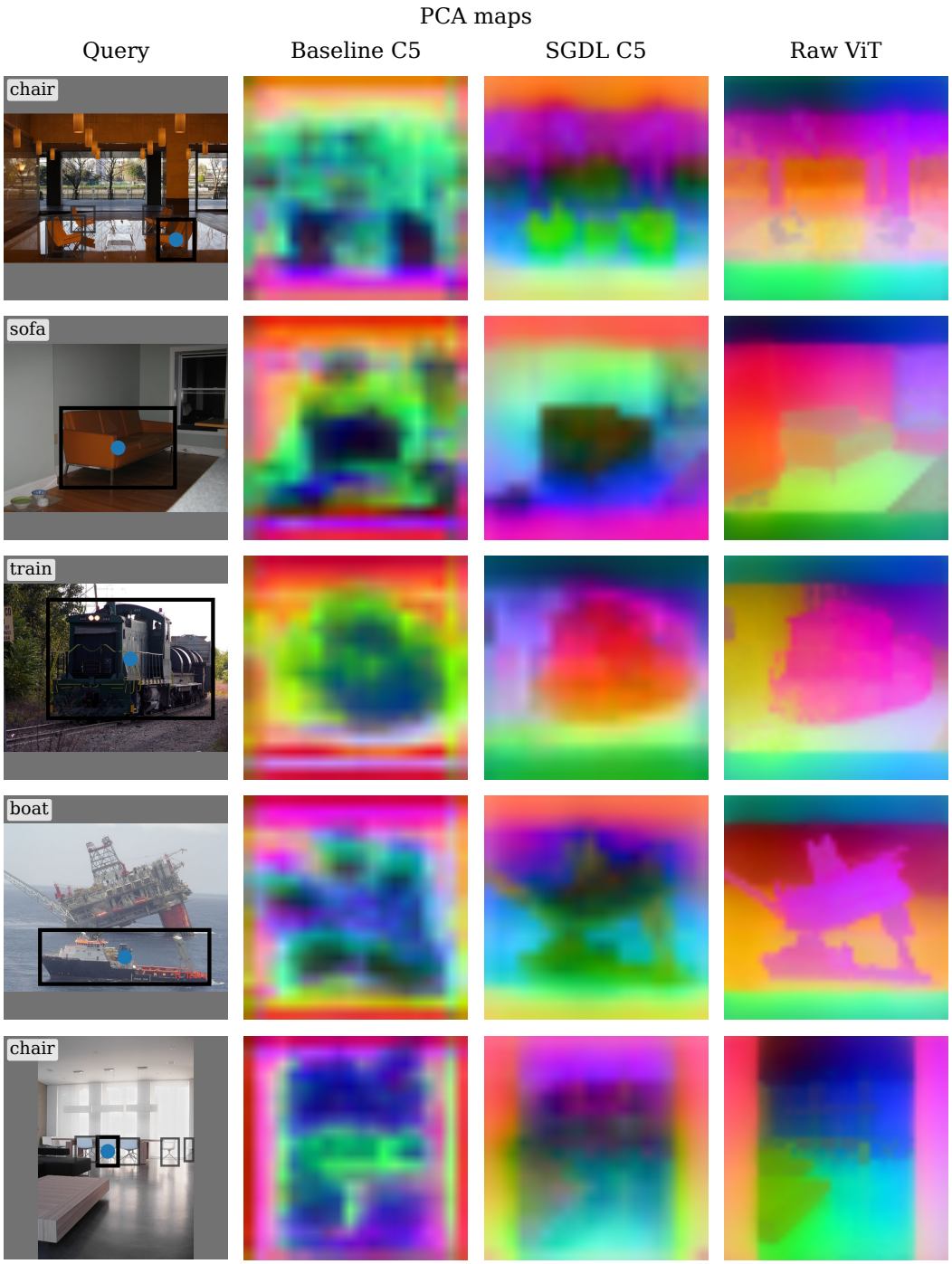

Figure 15: **Principal-component maps, continued,** for the images of Figure 12. Layout and colour mapping as in Figure 13.