SPLIT: STRUCTURE-PRESERVING BUDGET ALLOCATION FOR VISUAL ANOMALY DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Industrial anomaly detectors increasingly exploit multi-layer features from frozen backbones, but conventional layer-wise heads entangle detector budget with feature structure: a head's width is fixed by the backbone's feature dimension, so trainable budget can only be changed by changing which layers are consumed. We introduce **SPLIT** (**S**tructure-**P**reserving budget a**L**location for v**I**sual anomaly de**T**ection), a latent reparameterization of multi-layer anomaly heads whose latent dimension controls trainable budget at a fixed feature structure. Separating the two exposes a budget-dependent budget-sharing trade-off: under a matched total budget, a single shared latent code concentrates a scarce budget and wins at the smallest budgets, while per-layer codes remove cross-layer latent coupling at the cost of more budgets—take over once each head receives a workable width. The asymmetry between backbones is consistent with their cross-layer representation structure, and feature composition remains a third, independent factor. Across five industrial benchmarks and two frozen backbones, SPLIT is competitive with prior work, using only 0.002M latent parameters on DINOv3 and 0.008M on CLIP.

1 INTRODUCTION

Industrial visual anomaly detection must localize defects from a few tens of labeled images per class, and fine defects live in shallow texture statistics while structural anomalies live in deep semantics. Modern detectors (Gao & Wang, 2026; Yang, 2026) therefore freeze a large backbone (DINOv3 (Siméoni et al., 2025), CLIP (Radford et al., 2021)) and train only a layer-wise *segmentation head*—one 2×1024 probe per layer.

The budget–structure entanglement. With L layers that head has $L \times 2 \times 1024$ trainable parameters, reducible in exactly two ways: drop layers, or shrink the probe. The second is impossible—the probe width is pinned to the backbone's feature dimension—and the first is not a neutral budget reduction, because the layer set *is* the multi-scale structure. *How much trainable budget should the detector have?* and *which layers should it consume?* therefore cannot be answered independently: in Figure 1(a) the red arrow is the layer count, and moving it moves the parameter count and the consumed feature set at once. Note that "budget" denotes the number of trainable latent scalars throughout the paper. This is an operational parameter budget rather than a claim about intrinsic functional capacity. And we take per-layer projection as an example.

SPLIT: latent reparameterization of the head. We *generate* the head weights instead of training them: a frozen, shared projection $\mathbf{W} \in \mathbb{R}^{2D \times d}$ maps a trainable latent code $\mathbf{z}_l \in \mathbb{R}^d$ per layer to that layer's head, and the codes are the only trainable parameters. Because d does not touch the layer set, we obtain the property on which the rest of the paper is built:

> *Budget can be adjusted through the latent dimension d without discarding any feature layer—and the preferred regime is strongly backbone-dependent rather than fixed by the task.*

This is not unlimited freedom—a badly chosen d hurts—but it makes budget and feature structure separately observable. The reparameterization also has an *allocation* dimension: one code per layer costs $C{=}Ld$ and lets each layer follow its own code, whereas a single shared code spends $C{=}d_s$.

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.

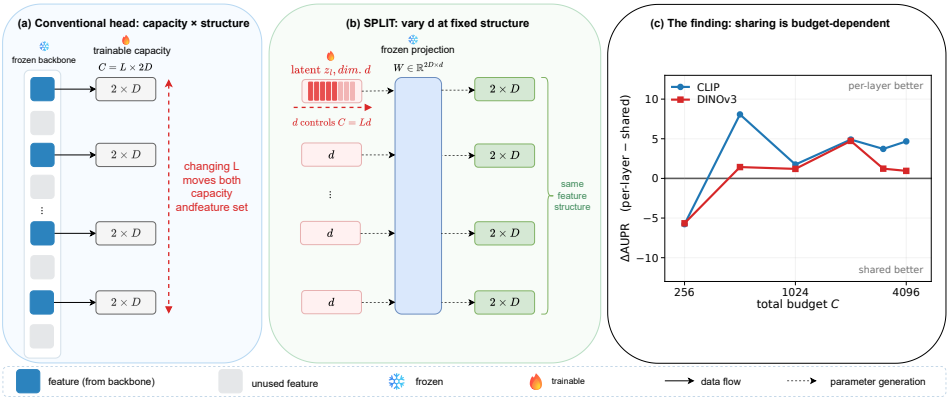

Figure 1: **Why SPLIT exists, and what it reveals.** *(a)* A conventional layer-wise head: the probe width is pinned to the feature dimension, so changing the trainable budget $C{=}L \times 2D$ means changing the number of consumed layers—and the layer set *is* the multi-scale structure. *(b)* SPLIT keeps the feature structure and generates each layer's head from its own latent code $\mathbf{z}_l$ of dimension d through a frozen, shared projection $\mathbf{W}$; budget is now $C{=}Ld$, so d controls budget and leaves the structure untouched. *(c)* Matched-budget comparison (Table 2): sharing wins at the tightest budget, per-layer allocation past $C \approx 512$.

The main finding. Fixing C and varying how it is split yields a trade-off rather than a ranking. As shown in Figure 1(c), a shared code wins at the tightest budget we test ($C{=}256$: 14.8 vs. 9.0 pixel-AUPR on CLIP, 44.0 vs. 38.4 on DINOv3) while per-layer codes lead from $C \approx 512$ onward by $+1.8$ to $+8.1$: sharing concentrates a scarce budget into one usable width but couples the layers' updates, whereas per-layer codes remove that coupling at the price of dividing the same budget. Two results accompany this: the backbone asymmetry is consistent with cross-layer representation structure (CLIP is far more heterogeneous across depth; mean linear centered kernel alignment (CKA) (Kornblith et al., 2019) 0.477 vs. 0.845), reported as supporting evidence rather than causal identification; and feature *composition* is a third independent factor, since changing only which layers are used at matched count and width still moves accuracy substantially (Sec. 4.6).

Contributions: *(i)* a reparameterization making budget and feature structure separately controllable, with a preferred budget regime that is strongly backbone-dependent ($d \approx 2048$–4096 for CLIP, 512 for DINOv3); *(ii)* a budget-dependent budget-sharing trade-off at matched C; and *(iii)* evidence that allocation and composition are distinct design factors. We also report an end-to-end evaluation under one harness, a generator sensitivity study, the resulting deployment cost, and the post-hoc truncation method (Appendices A.1–A.9).

2 Related Work

2.1 Multi-layer Industrial Anomaly Detection

Feature-store methods (e.g., PatchCore (Roth et al., 2022)) compare query features against a bank of normal features, whose greedy coreset costs $O(N^2)$ at the patch level. WinCLIP (Jeong et al., 2023) first used CLIP (Radford et al., 2021)'s vision–language alignment for zero-shot localization with hand-crafted prompts and multi-scale windows, at the cost of an expensive windowed re-encoding; AnomalyCLIP (Zhou et al., 2024) replaced the prompts with learnable object-agnostic ones (5.52M parameters). UniADet (Gao & Wang, 2026) moved to language-agnostic pure-visual detection with a decoupled head and only ≈0.02M trainable parameters on frozen features, and LUMIN (Yang, 2026) cuts that to $10\times$ smaller at the cost of compressing away some layer-wise information.

More recent work keeps the frozen backbone but changes what is learned on top of it: LAFT (Yun et al., 2025) learns a language-guided feature transformation so user knowledge can be injected without touching the backbone, and MMAD (Jiang et al., 2025) asks whether multimodal LLMs can serve as industrial inspectors, finding them far from expert level. Both leave the parameterization of the head itself untouched, which is what SPLIT varies.

Where SPLIT sits relative to these matters. UniADet already gives each layer an independent head, so "use a separate representation per layer" is not what we contribute. What a directly trained head cannot do is vary its budget without changing which layers it consumes—the probe width is pinned to D, making the layer count the only knob, and the layer set *is* the multi-scale structure. Our contribution is the observation that *how much* budget the detector has and *where* it is allocated are two variables a layer-wise head fuses into one, plus a reparameterization that separates them.

2.2 Latent Parameterization and Weight Generation

SPLIT's generator belongs to the weight-generation family—HyperNetworks (Ha et al., 2017) generate target weights from a hypernetwork and a latent code, and Mapping Network (Sen & Mukherjee, 2026) modulates frozen basis weights—and we regard that mechanism as a *tool* rather than as the contribution, since a frozen projection plus a trainable code is a well-established linear parameterization. Recent work pushes it further: (Soro et al., 2025) sample task-conditioned weights from a latent diffusion model as a transfer initialization, and (Gu & Yeung, 2025) use foundation models to improve the hypernetworks behind implicit representations. Neither varies how the budget is *allocated* across a hierarchy of heads. A related line trains in a randomly projected subspace rather than generating weights (Li et al., 2018); there the subspace is a training convenience, whereas here the latent variable is the quantity whose *allocation* is studied. Our contribution is therefore not a new weight-generation primitive, but a controlled parameterization that exposes trainable budget and its cross-layer allocation as separable experimental variables.

2.3 Cross-layer Representations in Foundation Models

For dense tasks, self-supervised ViT features (e.g. DINOv3 (Siméoni et al., 2025)) encode dense semantics in mid-to-deep layers, while early layers are too local and final layers too global. We reproduce this for DINOv3 and observe the same mid-deep preference for CLIP, with a smaller penalty for violating it (the deep-minus-shallow margin is 1.4–$1.8\times$ smaller, Sec. 4.6). Beyond this ranking, the backbones differ far more in how *consistent* their layers are with one another, which we quantify with linear CKA (Sec. 4.5). Such block structure is not incidental to ViTs—Jacobs et al. (Jacobs et al., 2026) show it is block-diagonal and exploitable for depth compression—but we use it as the representation-level quantity, not as a compression recipe.

3 Method

3.1 Problem setup and latent reparameterization

A frozen backbone provides patch features $\{f_l\}_{l\in\mathcal{L}}$ at selected layers $\mathcal{L}$ (the *feature structure*), with $L = |\mathcal{L}|$ and $D{=}1024$. Four quantities must be kept distinct: $P{=}L \times 2D$, the parameter count of target multi-layer head $H_l \in \mathbb{R}^{2\times D}$; d and d_s, the per-layer and shared latent dimension respectively; and C, the total trainable latent parameter budget ($C{=}Ld$ per-layer, $C{=}d_s$ shared).

SPLIT instead *generates* the head (Figure 2). Take per-layer projection as an example: A frozen, orthogonally initialized (Saxe et al., 2013) projection $\mathbf{W} \in \mathbb{R}^{2D\times d}$ maps a trainable code $\mathbf{z}_l \in \mathbb{R}^d$ to the flattened head parameters $\theta_l \in \mathbb{R}^{2D}$, reshaped into the segmentation head $H_l \in \mathbb{R}^{2\times D}$. The projection is shared across layers and never updated, so the codes are the only trainable parameters, and the same $\mathcal{L}$ feeds the same heads. Stacking one code per layer,

$$\mathbf{Z} = [\mathbf{z}_1; \ldots; \mathbf{z}_L] \in \mathbb{R}^{L\times d}, \qquad \theta_l = \sigma\big(\widetilde{\mathbf{W}}_l\, \mathbf{z}_l\big), \tag{1}$$

with σ applied coordinate-wise and $\widetilde{\mathbf{W}}_l$ the (possibly modulated) projection for layer l.

3.2 Latent Allocation Across Layers

Both structures generate the same $L \cdot 2D$ head parameters; they differ in what the projection spans and hence in how many trainable parameters they spend.

Per-layer. One projection $\mathbf{W} \in \mathbb{R}^{2D\times d}$ is reused by every layer while each layer carries its own code, $\theta_l = g(\mathbf{W}, \mathbf{z}_l) \in \mathbb{R}^{2D}$, so $C{=}Ld$ and the frozen basis is $2D \times d$. All three instantiations of Sec. 3.3 take this form.

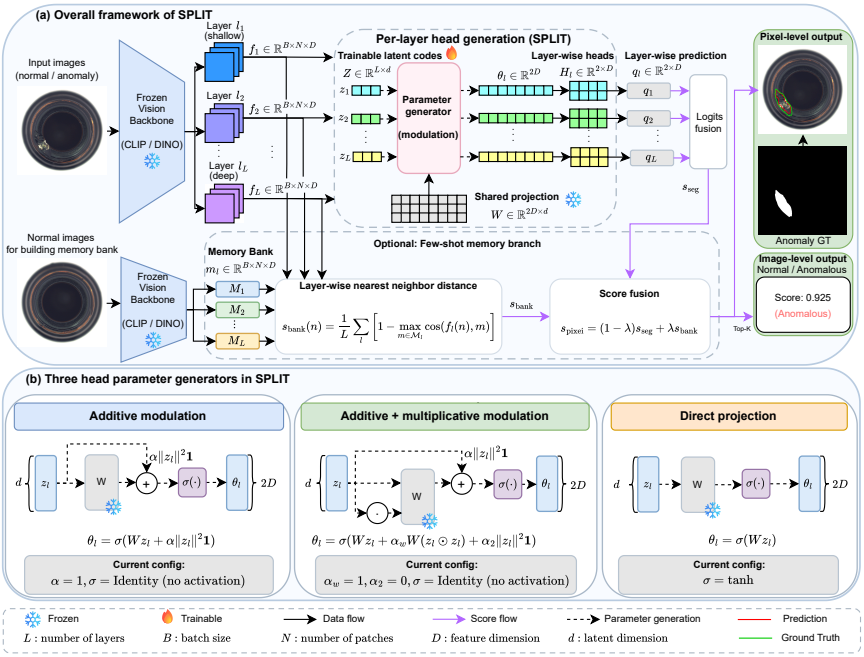

Figure 2: **SPLIT: architecture and generator instantiations.** *(a)* Overall framework. A frozen CLIP or DINOv3 backbone supplies multi-layer patch features $f_l \in \mathbb{R}^{B\times N\times D}$ (B: batch, N: patches); the only trainable parameters are the per-layer latent codes $\mathbf{Z} \in \mathbb{R}^{L\times d}$, which generate each layer's flattened head parameters $\theta_l \in \mathbb{R}^{2D}$ through a frozen, shared projection $\mathbf{W}$, reshaped into the segmentation head $H_l \in \mathbb{R}^{2\times D}$. A per-layer memory bank supplies the optional few-shot branch, whose score is fused with the segmentation score into the pixel map; Top-K pooling reduces that to an image-level score. *(b)* The three generator instantiations (Sec. 3.3) with the configuration used in the main experiments: additive (α=1, identity activation), additive–multiplicative (α_w=1, α_2=0, identity activation), and direct projection (tanh).

Shared. A single code $\mathbf{z}_s \in \mathbb{R}^{d_s}$ generates the *entire* set of head parameters at once, through one projection that spans all layers, $\mathbf{W}_s \in \mathbb{R}^{L\cdot 2D\times d_s}$: $\Theta = g(\mathbf{W}_s, \mathbf{z}_s) \in \mathbb{R}^{L\cdot 2D}$ with $\Theta = [\theta_1; \ldots; \theta_L]$, so $C{=}d_s$ and the frozen basis is $L \cdot 2D \times d_s$. The two parameterizations differ in how many codes there are and how much of the head each produces—one code for all L heads, versus one per head—and not in how a code is mapped. At a fixed nominal width they differ by a factor of L in budget and are comparable only once C is matched (Sec. 4.4).

What couples the layers under a shared code is best stated about the *gradient*. With per-layer codes the chain rule gives

$$\nabla_{\mathbf{z}_l}\mathcal{L} \;=\; J_l(\mathbf{z}_l)^\top \nabla_{\theta_l}\mathcal{L}, \qquad J_l(\mathbf{z}_l) \;=\; \frac{\partial \theta_l}{\partial \mathbf{z}_l}, \tag{2}$$

so layer l's gradient contribution enters $\mathbf{z}_l$ directly rather than being summed into a vector shared with other layers. Under a shared code the same contributions accumulate,

$$\nabla_{\mathbf{z}}\mathcal{L} \;=\; \sum_{l\in\mathcal{L}} J_l(\mathbf{z})^\top \nabla_{\theta_l}\mathcal{L}, \tag{3}$$

so one code must express the projected gradients of L objectives through a single update—not an infeasibility, and not necessarily a conflict, since the summed contributions could also reinforce each other.

3.3 Generator Instantiations

We instantiate $\widetilde{\mathbf{W}}_l$ in three ways, all sharing the same frozen $\mathbf{W}$ and code $\mathbf{z}_l$. They do not differ in modulation alone—SPLIT-D uses a $\tanh$ output nonlinearity where the other two use the identity—so they are best read as alternative *generator parameterizations* rather than as isolated modulation

ablations, and we treat the family as a *sensitivity study*, not a contribution. Truncating a large code instead of training a small one was also tried and dropped (Appendix A.7).

SPLIT (additive). $\widetilde{W}_{ij} = W_{ij} + \alpha \mathbf{z}_{l,j}$, i.e. $\widetilde{\mathbf{W}} = \mathbf{W} + \alpha \mathbf{1}\mathbf{z}^\top$ *column-wise* (j indexes the latent coordinate, not the output coordinate i), so that

$$\theta_l = \sigma\big(\mathbf{W}\mathbf{z}_l + \alpha \|\mathbf{z}_l\|^2 \, \mathbf{1}\big). \tag{4}$$

SPLIT-M (multiplicative). $\widetilde{W}_{ij} = W_{ij}(1 + \alpha_w \mathbf{z}_{l,j})$, so that

$$\theta_l = \sigma\big(\mathbf{W}\mathbf{z}_l + \alpha_w \, \mathbf{W}(\mathbf{z}_l \odot \mathbf{z}_l)\big), \tag{5}$$

each output coordinate being corrected by the projection of the squared code.

SPLIT-D (direct). No modulation: $\theta_l = \sigma(\mathbf{W}\mathbf{z}_l)$.

Configuration. All results use $\alpha{=}\alpha_w{=}1$; σ is the identity for SPLIT and SPLIT-M and $\tanh$ for SPLIT-D. Figure 2(b) draws SPLIT-M in a more general additive–multiplicative form carrying an extra $\alpha_2\|\mathbf{z}_l\|^2\mathbf{1}$ term; an ablation on α_2 found only marginal benefit from that branch, so we set $\alpha_2{=}0$ for simplicity and it reduces to Eq. 5.

The additive term is a null direction of the objective. Under the identity activation of Eq. 4, the additive term adds the same scalar $\alpha\|\mathbf{z}_l\|^2$ to *every* entry of θ_l, hence the same offset to both class logits at every patch. A shift shared by all classes leaves the softmax unchanged, so it changes neither the prediction nor the loss, and $\partial\mathcal{L}/\partial\alpha = 0$: SPLIT is *functionally equivalent* to a plain projection with identity activation, and α is an inert parameter. We keep the variant because it documents this flat direction, and because it is a concrete instance of the gap the paper is about. The practical consequence is that what separates SPLIT-D from the other two is its $\tanh$, and what separates SPLIT-M from SPLIT is its multiplicative term.

3.4 Anomaly Scoring and Few-Shot Fusion

What is trained, and when. Training never touches the memory bank: the backbone, $\mathbf{W}$ and all structural parameters are frozen and the optimizer updates only the latent codes $\mathbf{Z}$, the loss coming from the segmentation output alone. Evaluation freezes the codes too, so the only difference between the zero-shot and few-shot settings is whether a bank is built and fused in (Figure 2a).

Scores. The layer-wise logits are combined and passed through a softmax; $s_{\text{seg}}(n)$ is the resulting class-1 probability at patch n. For $k \geq 1$ we sample k normal images per class, build one bank M_l per consumed layer, and score a test patch by its distance to the nearest stored feature under cosine similarity:

$$s_{\text{bank}}(n) \;=\; \frac{1}{L}\sum_{l=1}^{L}\Big[\,1 - \max_{m \in M_l} \cos\big(f_l(n),\, m\big)\Big], \tag{6}$$

so a patch is anomalous to the degree that no normal feature from any layer is close to it. The two maps are then combined with a single fixed weight,

$$s_{\text{pixel}}(n) = (1-\lambda)\, s_{\text{seg}}(n) + \lambda\, s_{\text{bank}}(n), \tag{7}$$

with $\lambda{=}0.5$; at $k{=}0$ the second term is absent. Both terms are bounded (s_{seg} a softmax probability, s_{bank} a cosine distance) and we apply no further calibration. Image-level scores come from Top-K pooling (inference only, no trainable parameters).

4 Experiments and Discussion

The experiments are organized around five questions rather than around components: **Q1** does the detector work (Sec. 4.2); **Q2** can budget be varied at a fixed feature structure (Sec. 4.3); **Q3** under a fixed total budget, how should it be allocated across layers (Sec. 4.4); **Q4** does cross-layer representation structure track the backbone asymmetry (Sec. 4.5); **Q5** does feature composition matter once budget is matched (Sec. 4.6). Each varies one factor at a time—fix the structure and vary d; fix C and vary its allocation; fix L and d and vary which layers are used.

4.1 PROTOCOL

Benchmarks and backbones. We compare against AnomalyCLIP (Zhou et al., 2024), VisualAD (Hou et al., 2026), UniADet (Gao & Wang, 2026) and LUMIN (Yang, 2026) on MVTec-AD (Bergmann et al., 2019), VisA (Zou et al., 2022), BTAD (Mishra et al., 2021), KSDD (Tabernik et al., 2019) and Real-IAD (Wang et al., 2024), with frozen CLIP ViT-L/14@336px (Radford et al., 2021) and DINOv3 ViT-L/16 (Siméoni et al., 2025) backbones reading layers $[5, 11, 17, 23]$ (0-indexed).

Protocol. We follow the cross-dataset setting standard in CLIP-based anomaly detection (Gao & Wang, 2026; Yang, 2026): the detector is trained on the labeled split of a source benchmark conventionally designated as its *test* split (VisA for the MVTec/BTAD/KSDD/Real-IAD rows, MVTec for the VisA rows) and evaluated on disjoint target benchmarks without adaptation, DINOv3 under the identical protocol rather than a backbone-specific one. "Zero-shot" therefore means *zero target-domain adaptation*, not unseen classes. We use per-layer codes, Focal (Lin et al., 2017) plus Dice (Milletari et al., 2016) losses, AdamW (Loshchilov & Hutter, 2017) with a cosine-annealed rate (Loshchilov & Hutter, 2016) (lr 3×10^{-3} for ours and 1×10^{-3} for baselines), 19 epochs for ours and 15 epochs for baselines, and $k \in \{0, 1, 2\}$ on one NVIDIA L40S; metrics are image- and pixel-level AUROC, AUPR (Davis & Goadrich, 2006) and F1-max (Lipton et al., 2014).

The headline table uses a single fixed configuration—additive SPLIT, window $[5, 11, 17, 23]$ inherited from the baseline protocol rather than our window sweep, d=512 (DINOv3) / 2048 (CLIP)—with no per-dataset, per-k or per-metric selection: the per-backbone widths are fixed once and applied unchanged to all five benchmarks, and their transfer is itself part of the budget result (Sec. A.2).

4.2 Q1: MAIN CROSS-DOMAIN RESULTS

Table 1 reports the zero-shot comparison under that configuration. SPLIT is competitive with the strongest baseline in most cells—leading at pixel level on KSDD and on several MVTec and BTAD cells, while trailing AnomalyCLIP on MVTec and VisA and UniADet on BTAD and Real-IAD—so we describe it as *competitive* rather than state-of-the-art, with its strongest showing at pixel-level localization. SPLIT trains 0.002M latent parameters on DINOv3 and 0.008M on CLIP, compared with approximately 0.02M total trainable parameters in UniADet under our implementation as shown in Table 13. The full few-shot comparison is in Table 18 (Appendix A.9); a single normal sample already produces a step change over zero-shot, and further shots add little.

4.3 Q2: BUDGET CONTROL AT A FIXED FEATURE STRUCTURE

Figure 3 sweeps d with the window fixed at $[5, 11, 17, 23]$, so the consumed features are identical at every point and only the parameterization changes. CLIP's pixel-AUPR rises monotonically from 9.0 at d=64 to 41.7 at d=4096 and is still climbing, whereas DINOv3 peaks much earlier (50.0 at d=512 zero-shot) and is flat-to-decreasing beyond, consistently across all three generators. Since the structure is fixed throughout, this is a property of the parameterization rather than of feature selection.

The reading is bounded: d is not free, and choosing it badly is costly. What the result establishes is that the preferred regime is backbone-dependent—it moves neither with the training-set size (over a 20× range the argmax drifts from 512 to 1024 but never significantly, while the cost of choosing badly grows from 3.2 to 11.4 points; Appendix A.1) nor with the window (Appendix A.3).

4.4 Q3: BUDGET SHARING UNDER A MATCHED BUDGET

The comparison one would naturally run—shared versus per-layer at the same nominal d—is not parameter-matched: a shared code costs C=d against C=Ld, so at L=4 the per-layer model has 4× the budget and any difference conflates allocation with budget. Re-pairing them at matched C ($d_{shared} = 4\, d_{per\text{-}layer}$, Table 2) changes the picture: per-layer codes are *not* unconditionally better. At C=256 (64 dimensions per head) the shared code wins on both backbones (14.8 vs. 9.0 on CLIP, 44.0 vs. 38.4 on DINOv3) because splitting 256 across four heads leaves too little width each; per-layer codes lead from $C \approx 512$ onward, but by a much smaller margin (+1.8 to +8.1) than the fixed-d view suggests.

Table 1: Zero-shot (k=0) comparison against prior methods. Each cell reports image-level / pixel-level AUROC, AUPR, F1-max. AnomalyCLIP and VisualAD use CLIP; UniADet, LUMIN and SPLIT report both CLIP and DINOv3 backbones. All SPLIT cells use the *same* configuration—additive generator, fixed window $[5, 11, 17, 23]$, d=512 (DINOv3) / 2048 (CLIP)—with no per-dataset or per-metric selection (Sec. 4.1). **Red** = best value in each row segment (image-level or pixel-level), **blue** = second best.

Dataset	Metric	AnomCLIP	VisualAD	UniADet		LUMIN		SPLIT	
		CLIP	CLIP	CLIP	DINOv3	CLIP	DINOv3	CLIP	DINOv3
MVTec	AUROC	**91.6**/**91.1**	88.1/89.8	**89.7**/90.0	**91.6**/90.4	56.3/78.6	87.4/89.3	88.2/89.5	84.5/**90.7**
	AUPR	**96.4**/34.7	94.8/40.1	95.2/43.5	**96.2**/46.7	79.0/12.2	94.2/**47.7**	94.8/40.8	93.3/**50.0**
	F1max	**92.8**/39.3	91.4/42.6	91.6/44.8	**93.3**/47.2	84.6/18.0	88.8/**47.7**	90.8/42.6	88.2/**49.0**
VisA	AUROC	82.3/**96.1**	47.7/83.9	**86.9**/94.2	**89.8**/95.6	58.4/80.5	79.8/95.6	82.3/94.2	80.1/**96.0**
	AUPR	54.9/17.0	17.0/2.8	**61.3**/21.9	**69.6**/**25.1**	21.5/0.9	53.6/22.2	55.8/18.7	54.4/**24.3**
	F1max	54.4/23.6	25.8/4.2	**58.9**/26.4	**66.9**/**30.1**	28.5/2.0	52.6/27.0	54.0/24.0	53.4/**29.1**
BTAD	AUROC	89.1/93.4	88.4/86.3	**94.3**/**94.4**	90.9/**96.7**	62.4/64.7	**95.1**/**94.4**	92.5/93.3	90.1/94.3
	AUPR	91.0/42.0	91.6/31.2	**94.7**/45.3	89.9/**54.6**	75.5/3.9	93.5/**50.9**	**96.5**/40.8	85.8/45.2
	F1max	86.0/47.7	87.9/36.4	90.0/45.8	86.8/**53.9**	76.9/7.8	**92.7**/**51.6**	**92.4**/45.4	82.5/45.2
KSDD	AUROC	**97.0**/97.5	66.7/67.3	**97.9**/96.8	84.8/97.6	42.5/90.0	80.2/**97.7**	95.2/97.0	75.0/**97.8**
	AUPR	**84.7**/**19.7**	19.5/0.3	**85.0**/13.5	49.9/10.2	12.5/2.7	36.8/14.3	79.7/**24.3**	35.6/12.4
	F1max	**80.4**/**34.8**	34.2/1.8	**84.4**/25.3	57.1/24.6	23.0/7.5	44.4/26.4	76.7/**36.4**	41.6/22.9
Real-IAD	AUROC	68.2/93.0	**72.8**/94.8	66.2/**96.2**	67.9/**96.4**	52.5/81.5	68.3/95.5	**69.6**/96.1	64.6/94.9
	AUPR	54.3/23.8	**83.2**/27.0	76.9/35.6	79.2/**40.8**	67.4/1.1	79.9/**39.0**	**80.1**/34.8	77.4/35.8
	F1max	56.5/31.7	**81.2**/33.0	79.5/39.5	**79.9**/**43.6**	78.4/2.8	79.6/**42.5**	**79.9**/39.2	79.3/39.3

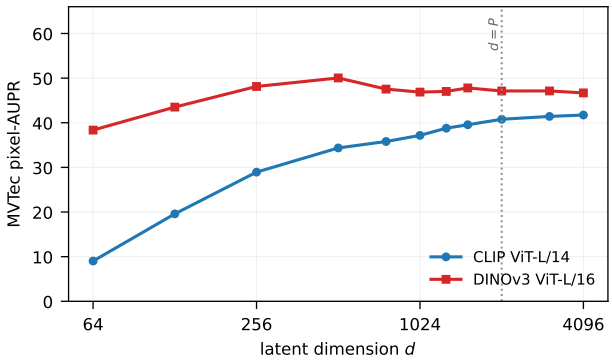

Figure 3: **Budget can be swept at a fixed feature structure.** Zero-shot (k=0) MVTec pixel-AUPR of the additive generator as the latent dimension d varies, with the feature window held at $[5, 11, 17, 23]$ throughout, so every point of a curve consumes the same backbone features and only the parameterization changes. CLIP keeps improving to the largest dimension tested (d=4096, 41.7), whereas DINOv3 peaks at d=512 (50.0) and flattens beyond it. The dotted line marks d=P=2048, where the latent dimension equals the parameter count of a single head; beyond it the parameterization is overcomplete and the curves gain little. The preferred regime is backbone-dependent rather than fixed by the feature structure or the data budget.

The reading is a trade-off rather than a ranking: sharing *concentrates* a scarce budget into one width every head can use, while per-layer allocation *removes the cross-layer coupling* of Eqs. 2–3.

4.5 Q4: Cross-layer representation structure

If the cost of sharing depends on how similar the per-layer targets are (Sec. 3.2), DINOv3's consistent layers should let one code approximate them all, whereas heterogeneous CLIP would force a compromise. Figure 4 quantifies that with linear CKA: DINOv3's matrix is uniformly bright (mean 0.845), whereas CLIP's has a clear block structure—layers 0–9, 10–17 and 18–23 each internally consistent, cross-block similarity near zero (0.477 mean, 0.203 between the extreme layers).

Table 2: **Budget sharing under a matched total budget** (zero-shot, additive generator, same 4-layer window). A shared code carries d_1 trainable parameters and a per-layer code carries $L \cdot d_2$, so the two are matched by $d_1 = 4d_2$; each row is one total budget and the per-layer column reports d_2. Sharing wins at the tightest budget on both backbones; per-layer allocation leads from ≈ 512 onward, by a far smaller margin than the fixed-d comparison implies.

		CLIP			DINO		
total cap. (d_1)	d_2	shared	per-layer	Δ	shared	per-layer	Δ
256	64	14.79	9.02	-5.77	44.02	38.36	-5.66
512	128	11.52	19.59	+8.07	42.08	43.51	+1.43
1024	256	27.19	28.94	+1.75	46.93	48.13	+1.20
2048	512	29.46	34.36	+4.90	45.31	50.04	+4.73
3072	768	32.08	35.80	+3.72	46.31	47.54	+1.23
4096	1024	32.50	37.17	+4.67	45.92	46.87	+0.95

Adjacent-layer similarity is high for both (0.987 vs. 0.947), so they differ in how *far* consistency extends, not whether it exists locally.

This matches CLIP's greater sensitivity to how budget is shared. Consistency and composition measure different aspects of the backbone—CKA contextualizes the sharing sensitivity, while the window ablation of Sec. 4.6 measures the utility of feature depth—and we keep the two claims separate, reporting the correspondence as supporting evidence rather than identification (two backbones, observational; geometry in Figure 5, Appendix A.8).

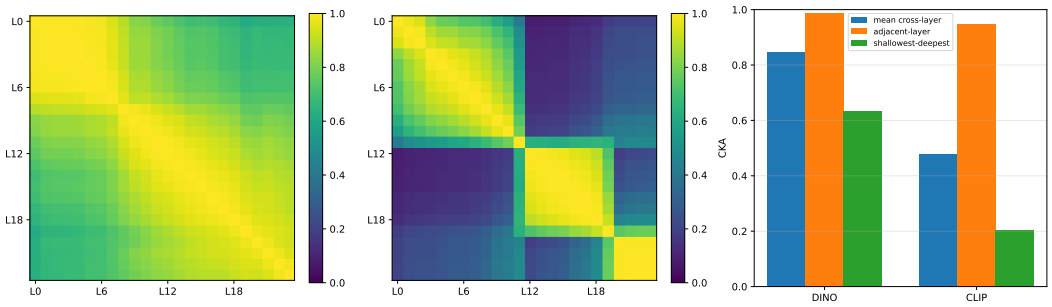

Figure 4: **Cross-layer representation consistency.** Left: inter-layer linear CKA of DINOv3 ViT-L/16 across layers 0–23. Middle: the same for CLIP ViT-L/14. Right: summary of three CKA statistics (mean cross-layer, adjacent-layer, and shallowest–deepest) for both backbones. Brighter is more similar. DINOv3 is uniformly consistent (0.845 mean); CLIP decomposes into blocks with near-zero cross-block similarity (0.477 mean). This is the representation-level contrast with the backbone-dependent sharing sensitivity observed in Sec. 4.4.

4.6 Q5: Feature composition at matched budget

Finally, is the feature structure its own design variable once budget is matched? We hold L and d fixed and vary only *which* layers the head consumes, comparing windows of *equal* size so that only composition varies within a group (full sweep of 32 layer sets in Table 9).

Composition clearly matters: at every matched size a deep window beats a shallow-anchored one of the same size, the margin shrinking from $+17.7$ (L=1) to $+4.5$ (L=4) on CLIP and $+29.7$ to $+6.4$ on DINOv3. Since both arms share the same L *and* d, this cannot be attributed to budget. The best window is a mid-deep 4–6-layer set for five of six (generator, backbone) combinations—within the ViT-L backbones and benchmarks studied here—and the deep-minus-shallow margin is consistently 1.4–$1.8\times$ larger on DINOv3 than on CLIP, indicating that feature-depth selection is more consequential for DINOv3 in this sweep. The shallow-layer penalty is not uniform, however: layer 2 is consistently detrimental while layer 5 helps on DINOv3 and layer 8 is near-neutral, so the blanket claim that shallow layers are harmful does not survive (Table 10, Appendix A.3).

5 Conclusion and Limitations

We presented a latent reparameterization that separates three design choices in cross-domain anomaly detection: the trainable latent budget, how that budget is allocated across backbone depths, and which features are consumed. Under a matched total budget, sharing is preferable at the smallest evaluated budget, while per-layer allocation becomes competitive or superior as the budget increases. The magnitude of this effect differs between CLIP and DINOv3 and is consistent with their cross-layer representation similarity. Separately, matched-budget window ablations show that feature composition remains important. These results motivate treating trainable budget, allocation, and feature structure as distinct design axes rather than conflating them through the number of layer-wise heads.

Limitations. The link between sharing behaviour and representation structure is observational across two backbones—a third backbone family would be the natural test—and the two differ in more than cross-layer structure (objective, patch size, resolution), so whether the trade-off transfers to CNNs, other ViT scales or other dense tasks is open. **Future work:** the projection is frozen after orthogonal initialization, so whether modulation, a different initialization, or layer-specific projections change the picture is open.

AI use statement

We used generative AI tools to assist with drafting and polishing the text. We did not use them for data collection, analysis, results or figures, and no other required disclosure applies. We reviewed all AI-assisted content and take full responsibility for the final submission.

Reproducibility statement

Model parameterizations and scoring are specified in Secs. 3.1–3.4, the main training and evaluation protocol in Sec. 4.1, and ablation-specific settings in Appendix A. We will release code and configuration files upon publication.

References

Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad—a comprehensive real-world dataset for unsupervised anomaly detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9584–9592. IEEE, 2019.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, 2006.

Bin-Bin Gao and Chengjie Wang. One language-free foundation model is enough for universal vision anomaly detection. *arXiv preprint arXiv:2601.05552*, 2026.

Jeffrey Gu and Serena Yeung. Foundation models secretly understand neural network weights: Enhancing hypernetwork architectures with foundation models. In *International Conference on Learning Representations*, volume 2025, pp. 58565–58577, 2025.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkpACe1lx.

Yanning Hou, Peiyuan Li, Zirui Liu, Yitong Wang, Yanran Ruan, Jianfeng Qiu, and Ke Xu. Visualad: Language-free zero-shot anomaly detection via vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21346–21356, June 2026.

Mozes Jacobs, Thomas Fel, Richard Hakim, Alessandra Brondetta, Demba Ba, and T Anderson Keller. Block recurrent dynamics in vision transformers. In *International Conference on Learning Representations*, volume 2026, pp. 51617–51642, 2026.

Jongheon Jeong, Yang Zou, Taewan Kim, Dongqing Zhang, Avinash Ravichandran, and Onkar Dabeer. Winclip: Zero-/few-shot anomaly classification and segmentation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19606–19616. IEEE, 2023.

Xi Jiang, Jian Li, Hanqiu Deng, Yong Liu, Bin-Bin Gao, Yifeng Zhou, Jialin Li, Chengjie Wang, and Feng Zheng. Mmad: A comprehensive benchmark for multimodal large language models in industrial anomaly detection. In *International conference on learning representations*, volume 2025, pp. 87273–87295, 2025.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy. Thresholding classifiers to maximize f1 score. *arXiv preprint arXiv:1402.1892*, 2014.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pp. 565–571. Ieee, 2016.

Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. Vt-adl: A vision transformer network for image anomaly detection and localization. In *2021 IEEE 30th international symposium on industrial electronics (ISIE)*, pp. 01–06. IEEE, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14298–14308. IEEE, 2022.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Lord Sen and Shyamapada Mukherjee. Mapping networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 36215–36223, June 2026.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Bedionita Soro, Bruno Andreis, Hayeon Lee, Wonyong Jeong, Song Chong, Frank Hutter, and Sung Ju Hwang. Diffusion-based neural network weights generation. In *International Conference on Learning Representations*, volume 2025, pp. 99644–99675, 2025.

Domen Tabernik, Samo Šela, Jure Skvarč, and Danijel Skočaj. Segmentation-Based Deep-Learning Approach for Surface-Defect Detection. *Journal of Intelligent Manufacturing*, May 2019. ISSN 1572-8145. doi: 10.1007/s10845-019-01476-x.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Chengjie Wang, Wenbing Zhu, Bin-Bin Gao, Zhenye Gan, Jiangning Zhang, Zhihao Gu, Shuguang Qian, Mingang Chen, and Lizhuang Ma. Real-iad: A real-world multi-view dataset for benchmarking versatile industrial anomaly detection. In *2024 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 22883–22892. IEEE, 2024.

Pengfei Yang. Lumin: Lightweight universal manufacturing inspection network for anomaly detection, 2026. URL https://arxiv.org/abs/2609.04775.

EungGu Yun, Heonjin Ha, Yeongwoo Nam, and Bryan Lee. Language-assisted feature transformation for anomaly detection. In *International Conference on Learning Representations*, volume 2025, pp. 15204–15224, 2025.

Qihang Zhou, Guansong Pang, Yu Tian, Shibo He, and Jiming Chen. Anomalyclip: Object-agnostic prompt learning for zero-shot anomaly detection. In *International Conference on Learning Representations*, volume 2024, pp. 49705–49737, 2024.

Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. In *European conference on computer vision*, pp. 392–408. Springer, 2022.

A Additional ablations

This appendix holds the full grids behind the main-text claims, grouped as the budget and feature-window studies, the generator study, the cost and compression measurements, and the supporting evidence.

A.1 Latent-width sweeps and backbone-dependent regimes

This section documents the budget sweeps behind Secs. 4.3 and 4.4. Its companions are Appendix A.2 for the shared-versus-per-layer comparison and Appendix A.3 for how these quantities behave when the layer window changes.

Full dimension sweeps. Table 3 gives the complete sweep behind the dimension curves of Sec. 4.3: MVTec pixel-AUPR at $k{=}0$ for each of the three generators. The backbone split is visible for all three—the CLIP curve is still rising at $d{=}4096$, while the DINOv3 curve has already passed its 512 peak—and the ordering of the generators differs between backbones, which is the sensitivity studied in Sec. A.4.

Table 3: Full latent-dimension sweep for all three generators, MVTec pixel-AUPR at $k{=}0$ (DINOv3, visA$\rightarrow$MVTec, per-layer codes). Columns are grouped by backbone; within each group the three generator instantiations are compared at the same width, so the sweep isolates budget from the feature structure.

	CLIP			DINOv3		
d	SPLIT	SPLIT-M	SPLIT-D	SPLIT	SPLIT-M	SPLIT-D
64	9.02	21.44	11.70	38.36	36.08	37.83
128	19.59	28.34	19.56	43.51	38.57	41.51
256	28.94	31.41	26.93	48.13	44.14	49.48
512	34.36	39.12	32.62	50.04	49.07	50.88
768	35.80	39.20	34.37	47.54	47.32	49.60
1024	37.17	40.28	35.44	46.87	46.16	47.87
1280	38.78	42.33	36.56	47.02	45.68	47.01
1536	39.55	41.86	37.62	47.80	47.82	47.58
2048	40.78	43.26	39.38	47.12	47.25	47.63
3072	41.42	43.14	40.24	47.13	46.89	46.92
4096	41.73	43.73	40.80	46.69	46.13	46.68

The dimension optimum is not a function of the data budget. Sec. 4.3 picks d by backbone. To test whether that choice is instead a function of how much data is available, we re-sweep d at five training-set fractions (VisA per-category samples scaled by the fraction; DINOv3, per-layer latent, k=0, additive). The optimum does move, but not significantly—at fraction 0.05 the best d (1024) and d=512 differ by 0.05 points (p=0.968, paired t-test over the 15 MVTec categories), and at 0.1 by 1.64 points (p=0.076). What does change is the *cost of choosing badly*: the spread across d grows from 3.2 points at full data to 11.4 at fraction 0.05, driven almost entirely by $d < 512$ (-11.3 points for d=384 at fraction 0.05, $p < 0.001$). Budget therefore matters more when data is scarce, but its optimum does not need to be re-tuned—a fact we rely on in Sec. 4.3. The full grid is in Table 4.

Table 4: Dimension versus training-set fraction. MVTec pixel-AUPR (DINOv3, per-layer latent, k=0, additive) for each d and each fraction of the VisA training set. The bottom row gives the argmax. At full data the best evaluated width is d=512; as data shrinks the preferred regime widens and drifts to 1024, but never significantly ($p \geq 0.076$), while $d < 512$ degrades sharply (-11.3 points for d=384 at fraction 0.05). The knob thus matters more under scarcity without needing to be re-tuned.

	training-set fraction					
d	1.0	0.75	0.5	0.25	0.1	0.05
256	48.15	48.25	46.63	44.23	41.66	32.31
384	47.28	48.05	46.54	45.41	42.03	30.69
512	**50.08**	**50.97**	**49.65**	47.60	45.15	42.01
640	48.56	49.18	48.08	48.03	45.08	40.73
768	47.52	48.27	48.22	**48.15**	46.16	41.45
1024	46.90	47.84	47.00	47.45	**46.78**	**42.06**
argmax	512	512	512	768	1024	1024

A.2 Latent structure: shared versus per-layer

Every sweep above uses per-layer codes. This section compares that structure against a single shared code: first as the layer count grows, then at fixed and at matched budget. The generator is held fixed throughout, so the comparison isolates the latent structure.

More layers, and what the 8-layer case does and does not show. Table 5 compares the two shared-latent runs that differ only in layer count. At 8 layers $[2, 5, 8, 11, 14, 17, 20, 23]$ the shared code peaks at d=768 (10.9) and has fallen to 5.9 by d=2048, whereas at L=4 the same generator climbs past 27 beyond d=1024—a gap of 19 to 25 points. The gradient view of Eq. 3 makes such a degradation plausible—a single code absorbing projected gradients from more objectives has more room to be pulled in different directions—but this comparison cannot establish that reading on its own: the 8-layer configuration also includes shallow layers $2/5/8$, and layer 2 is independently harmful for both backbones (Sec. 4.6). It is an observational comparison across two complete configurations, not a single-variable ablation, and we report it as such.

The fixed-d view, for reference. Table 6 holds d fixed rather than matching the budget, so the per-layer arm carries L times the parameters of the shared arm; it illustrates the *shape* of the two curves rather than a fair allocation comparison, which is Table 2 in the main text. The asymmetry it shows is is consistent with the CKA contrast 4.5: on CLIP the shared code saturates at 32.5, 9.2 points below the per-layer 41.7 at the same nominal width, whereas on DINOv3 the two are indistinguishable for $d \geq 1024$ ($|\Delta| \leq 0.19$), the shared code losing only where it is budget-constrained rather than compromised.

What the two structures amount to. Sharing concentrates budget into one width every head can use but forces all layer-specific gradient contributions to be expressed through a common update (Eq. 3); per-layer codes solve this problem but leave each head only d dimensions, so neither end of Table 2 is the general answer. The same reading explains the fixed-d view of Table 6, where the per-layer arm has $4\times$ the budget and most of the CLIP gap it shows is the mismatch rather than the coupling. The argument is not specific to anomaly detection: any multi-head detector whose heads come from one latent code inherits it.

Table 5: Shared-latent runs differing only in layer count, zero-shot (k=0; CLIP backbone, additive modulation, mean fusion, no regularization, seed 42, lr 3×10^{-3}, 19 epochs; 15-category mean pixel-AUPR). The 4-layer run is $[5, 11, 17, 23]$ and the 8-layer run $[2, 5, 8, 11, 14, 17, 20, 23]$, so the comparison changes the layer count *and* admits shallow layers at once: it is observational across two complete configurations, not a single-variable ablation. The 8-layer run degrades as d grows while the 4-layer run improves.

d	L=4	L=8	Δ
512	11.52	8.60	+2.92
768	12.33	10.86	+1.47
1024	27.19	8.13	+19.06
1280	30.64	6.60	+24.04
1536	31.12	5.79	+25.33
2048	29.46	5.94	+23.52

Table 6: Fixed-d comparison (zero-shot pixel-AUPR, both backbones, same 4-layer window $[5, 11, 17, 23]$, additive generator); only the latent structure varies (one shared code $\mathbf{z}$ vs. per-layer codes $\mathbf{z}_l$). *This pairing is not parameter-matched*—a shared code carries d parameters and a per-layer code $L \cdot d$, as the total-budget column makes explicit—so it should be read for the shape of the curves; the matched-budget comparison is Table 2 in the main text.

		CLIP			DINO		
d	total cap. (shared / per-layer)	shared	per-layer	Δ	shared	per-layer	Δ
64	64 / 256	5.60	9.02	+3.42	26.56	38.36	+11.80
128	128 / 512	9.49	19.59	+10.10	36.52	43.51	+6.99
256	256 / 1024	14.79	28.94	+14.15	44.02	48.13	+4.11
512	512 / 2048	11.52	34.36	+22.84	42.08	50.04	+7.96
768	768 / 3072	12.33	35.80	+23.47	46.37	47.54	+1.17
1024	1024 / 4096	27.19	37.17	+9.98	46.93	46.87	-0.06
1280	1280 / 5120	30.64	38.78	+8.14	47.09	47.02	-0.07
1536	1536 / 6144	31.12	39.55	+8.43	47.99	47.80	-0.19
2048	2048 / 8192	29.46	40.78	+11.32	45.31	47.12	+1.81
3072	3072 / 12288	32.08	41.42	+9.34	46.31	47.13	+0.82
4096	4096 / 16384	32.50	41.73	+9.23	45.92	46.69	+0.77

A.3 FEATURE-WINDOW ABLATION

The dimension optimum versus the layer window. Holding the generator and d's search grid fixed and varying only *which* layers the head consumes, the per-layer code keeps the same optimum: d=512 for a 2-, 4-, 5- and 6-layer window alike, and the 5-layer all-deep window has the sharpest peak of any of them (d=512 beats 256 by $+5.1$, p=0.002, and 768 by $+4.0$) even though that window is twice as wide as the 2-layer one. The shared code behaves differently: its optimum tracks the total number of head parameters and drifts upward with the window ($768 \rightarrow 1536 \rightarrow 2048$ from 2 to 5 layers, Table 7). The reason is the one identified in Sec. 3.2—a per-layer code carries d dimensions *per head* and so answers to a fixed per-head target, whereas a shared code must serve all L heads at once and therefore needs a dimension that grows with the total it represents. Across the practical 2–6-layer windows studied here, the per-layer variant retains the same best-performing evaluated width, whereas the shared variant shows a stronger shift with window size. The 8-layer shallow-heavy configuration is an exception.

Table 7: Optimal latent dimension as a function of the feature window (DINOv3, additive, k=0, VisA→MVTec, 15-category mean pixel-AUPR). Each row comes from its own sweep—the grid is not the same across rows (11–13 points for $[5, 11, 17, 23]$, 5 for the five- and six-layer windows), so the argmax may fall between the points tabulated in Table 8, whose common five-point grid does not contain 1536. The preferred width stays at 512 across the practical 2–6-layer windows we evaluate (the 8-layer shallow-heavy set is an exception); the shared optimum grows with the window, consistent with a per-head versus total-parameter interpretation of the code's role.

Window	L	shallow?	per-layer argmax	shared argmax
$[11, 23]$	2	no	512	768
$[5, 11, 17, 23]$	4	5	512	1536
$[11, 14, 17, 20, 23]$	5	no	512	2048
$[8, 11, 14, 17, 20, 23]$	6	8	512	1024
$[2, 5, 8, 11, 14, 17, 20, 23]$	8	2,5,8	1024	2048

Dimension versus feature window (DINOv3, additive, k=0, pixel-AUPR). Table 8 gives the full curves behind Table 7: each panel fixes a layer window and sweeps d, with the per-layer and shared variants as separate models. Two details the summary above does not carry. The shared curve is *broad*—d=2048 vs. 512 at five layers is only $+1.5$ (n.s.)—so the shared optimum is a weak preference rather than a sharp one. And the per-layer invariance has one exception: in the full 8-layer window, 1024 edges 512 by $+1.2$ (n.s.), consistent with the single-layer attribution below that only layer 2 is uniformly harmful.

Table 8: Per-layer codes (top) and shared codes (bottom): MVTec pixel-AUPR on a common five-point grid $d \in \{256, 512, 768, 1024, 2048\}$ for five layer windows; bold marks each row's maximum on *this* grid. These are not the sweeps behind Table 7, which uses a wider grid for the two- and four-layer rows—hence the 1536 argmax reported there does not appear here.

per-layer code					
Window (L)	256	512	768	1024	2048
$[11, 23]$ (2)	43.92	**47.79**	45.52	47.33	—
$[5, 11, 17, 23]$ (4)	48.13	**50.04**	47.54	46.87	47.12
$[11, 14, 17, 20, 23]$ (5)	45.24	**50.33**	46.34	48.18	46.79
$[8, 11, 14, 17, 20, 23]$ (6)	49.16	**49.49**	49.48	48.68	47.54
$[2, 5, 8, 11, 14, 17, 20, 23]$ (8)	48.06	48.95	47.51	**50.11**	47.83
shared code					
Window (L)	256	512	768	1024	2048
$[11, 23]$ (2)	39.02	43.12	**45.59**	43.18	43.10
$[5, 11, 17, 23]$ (4)	44.02	42.08	46.37	46.93	45.31
$[11, 14, 17, 20, 23]$ (5)	44.32	46.48	46.97	46.40	**47.95**
$[8, 11, 14, 17, 20, 23]$ (6)	40.24	48.15	47.29	**48.66**	48.22
$[2, 5, 8, 11, 14, 17, 20, 23]$ (8)	38.25	42.66	44.96	47.68	**49.89**

Cells marked "—" were not swept; the L=2 run used a slightly different dim grid ($\{64, \ldots, 1024\}$ for the per-layer variant), which does not affect the comparison of interest here. The L=2 rows are the cleanest instance of the pattern: at two layers the per-layer code still peaks at 512 (47.79, beating 768 by +2.27) while the shared code already peaks at 768.

Feature-window ablation (DINOv3 and CLIP, k=0, pixel-AUPR). We fix the latent dimension at d=512 and vary only *which* layers the head consumes. Every row of Table 9 is a different layer set at the same d, so rows differing in composition but not in length are directly comparable; rows are grouped by layer count L. The best-performing window in our sweep is a mid-deep 4–6-layer set for five of the six (generator, backbone) combinations ($[11, 14, 17, 20, 23]$ or $[11, 14, 17, 20]$), and shallow-anchored sets (those containing layer 2, and to a lesser extent 5 and 8) are clearly worse than deep sets of the same size.

Table 9: Feature-window ablation at a fixed latent dimension (d=512), zero-shot (k=0), MVTec pixel-AUPR. Rows are grouped by the number of layers L (separated by rules). All entries in a group use the same number of layers, so within a group only layer *composition* varies. Bold is omitted so that every cell can be compared directly.

	CLIP			DINOv3		
Layer set	SPLIT	SPLIT-M	SPLIT-D	SPLIT	SPLIT-M	SPLIT-D
[11]	32.7	36.7	17.1	36.9	37.1	34.2
[14]	29.6	32.9	10.7	43.1	42.3	41.7
[17]	31.6	31.6	7.0	44.9	43.4	44.3
[20]	31.3	30.8	25.5	41.0	38.3	40.9
[23]	30.5	31.1	27.9	42.3	41.0	42.4
[2]	11.3	11.1	9.1	8.0	10.0	4.9
[5]	15.6	16.4	8.1	15.9	16.7	9.6
[11, 14]	36.5	36.9	20.6	42.9	42.6	40.6
[11, 17]	37.1	38.1	12.6	45.8	44.1	44.1
[11, 23]	33.9	38.7	29.0	47.8	45.4	47.0
[17, 20]	33.4	34.4	25.1	48.2	46.1	48.5
[17, 23]	33.0	36.0	28.4	48.6	46.3	48.5
[20, 23]	31.5	33.2	27.6	47.3	45.1	47.3
[2, 5]	13.9	15.4	10.9	14.2	15.8	8.1
[5, 11]	29.7	34.5	14.6	34.8	36.9	28.9
[11, 14, 17]	34.8	36.7	13.2	45.1	44.0	44.5
[11, 17, 23]	35.2	37.9	28.8	47.1	46.7	47.2
[14, 17, 20]	35.2	36.3	22.5	46.1	44.3	46.1
[17, 20, 23]	33.8	34.8	29.0	46.3	44.9	46.4
[2, 5, 8]	22.5	23.9	18.4	21.4	22.3	10.4
[5, 11, 17]	32.1	36.3	14.3	44.0	43.1	41.8
[11, 14, 17, 20]	37.5	41.5	21.9	46.6	44.7	46.8
[14, 17, 20, 23]	34.8	39.2	29.0	49.2	47.5	49.8
[2, 5, 8, 11]	28.9	34.4	21.2	32.9	35.5	23.9
[5, 11, 17, 23]	34.4	39.1	27.6	50.1	49.1	50.3
[11, 14, 17, 20, 23]	35.1	39.8	27.6	50.3	48.5	51.0
[2, 5, 8, 11, 14]	29.5	35.5	21.7	41.0	41.7	33.6
[2, 5, 8, 11, 14, 17]	32.2	38.5	22.7	44.1	42.6	41.0
[8, 11, 14, 17, 20, 23]	35.8	38.9	29.1	49.5	49.1	50.0
[2, 5, 8, 11, 14, 17, 20]	33.7	37.5	16.3	46.4	45.2	46.0
[5, 8, 11, 14, 17, 20, 23]	34.5	38.5	27.9	49.9	49.1	50.6
[2, 5, 8, 11, 14, 17, 20, 23]	33.8	38.0	26.9	48.9	48.2	49.1

Same-budget composition effect. Holding L fixed and averaging over the available deep and shallow-anchored sets, a deep window beats a shallow-anchored one of the same length at every size. The margin shrinks as the layer count grows—a single deep layer already beats a single shallow layer by 18–30 points, whereas four layers of the same size differ by only 1–11—and the DINOv3 margin is 1.4–1.8× the CLIP margin at every size. This is the controlled version of the "shallow

layers hurt" statement: because both arms of each row use the same L and the same d, the effect cannot be attributed to budget.

Table 10: Deep-minus-shallow margin in MVTec pixel-AUPR, holding the layer count L fixed and averaging over the available deep and shallow-anchored sets of that size (d=512, k=0). Every pair compared uses the same number of layers *and* the same latent dimension. The values aggregate the per-set results in Table 9, which is the underlying sweep; the margin is tabulated separately because it is not readable off that table.

	CLIP			DINOv3		
L	SPLIT	SPLIT-M	SPLIT-D	SPLIT	SPLIT-M	SPLIT-D
1	+17.7	+18.9	+9.1	+29.7	+27.1	+33.4
2	+12.5	+11.3	+11.1	+22.3	+18.6	+27.5
3	+7.4	+6.3	+7.0	+13.4	+12.3	+20.0
4	+4.5	+3.6	+1.0	+6.4	+3.8	+11.2
5	+5.6	+4.2	+5.9	+9.4	+6.9	+17.3

Single-layer attribution. Comparing layer sets that differ by exactly one layer isolates which shallow layers are responsible. Adding layer 2 to $[5, 8, 11, 14, 17, 20, 23]$ costs -0.6 to -1.4 in all six configurations. Layer 5 is *not* uniformly harmful: adding it to $[11, 17, 23]$ gains $+2.4$ to $+3.1$ on DINOv3 (and is mixed on CLIP), consistent with $[5, 11, 17, 23]$ being the second-best DINOv3 window in Table 9. Layer 8 is near-neutral ($|\Delta| \leq 1.4$, sign inconsistent). Mid-deep layers are the real anchors: adding layer 23 to $[11, 14, 17, 20]$ is worth $+3.7$ to $+4.2$ on DINOv3. The blanket statement "shallow layers ≤ 8 are harmful" therefore does not survive the controlled comparison—only layer 2 does.

A.4 Generator sensitivity

Per-(dataset,k) best generator. Table 11 counts, per (dataset, k), how often each generator wins, out of the twelve image/pixel $\times$ CLIP/DINOv3 $\times$ metric cells. The outcome is strongly dataset-dependent: SPLIT-M wins on MVTec and BTAD for all k ($34.5/48$, $35.5/48$), SPLIT-D on VisA and KSDD ($26/48$, $46/48$), and additive SPLIT on Real-IAD at $k \leq 2$. No generator dominates everywhere and the spread is small (k=0 pixel-level mean: 53.6 SPLIT-D, 54.8 additive, 54.7 SPLIT-M), so we read this as an instantiation detail rather than a result about anomaly detection. These counts are deliberately *not* used to select a generator for the headline table: they are computed on the target benchmarks, and selecting on them would make the main results uninterpretable as a fixed-configuration comparison (Sec. 4.1).

The three should indeed win on different data, though not for the reason the modulation geometry alone suggests: the additive axis is a null direction (Sec. 3.3), so what separates the configurations is the multiplicative term and the output nonlinearity. The sensitivity in Table 11 is therefore *generator parameterization* sensitivity rather than an effect of modulation (Sec. A.5).

A.5 Why the three generators should not be expected to tie

The three instantiations differ in what they let the columns of the projection do, which is an exact consequence of their definitions rather than an empirical claim. Writing $\widetilde{\mathbf{W}} = [\tilde{w}^{(1)}, \ldots, \tilde{w}^{(d)}]$, the pre-activation head $\widetilde{\mathbf{W}}\mathbf{z}$ is a combination of the columns with the code as coefficients, so the columns are the basis *directions* and $\mathbf{z}$ only reweights them. (The activation σ is applied afterwards and is a fixed coordinate-wise function—the identity for SPLIT and SPLIT-M, $\tanh$ for SPLIT-D—so it adds no directions of its own.) Then:

- **Additive (SPLIT).** $\tilde{w}^{(j)} = w^{(j)} + \alpha \mathbf{z}_j \mathbf{1}$: every column is displaced, so the basis *directions* move in parameter space. The displacement is along a single fixed axis $\mathbf{1}_P$ for all j, by an amount read off the code, so the directions can be turned only within the one-dimensional family that axis generates; each column's displacement depends on z_j alone, and no column is modified by another's code. This is a parameter-space statement only: the axis $\mathbf{1}_P$ is a null direction of the two-class objective (Sec. 3.3), so the displacement leaves the detector's output unchanged.

- **Multiplicative (SPLIT-M).** $\tilde{w}^{(j)} = (1 + \alpha_w \mathbf{z}_j)\, w^{(j)}$: the direction of each column is exactly preserved up to a possible sign flip — the prefactor can be negative, so the column stays on the same line but may reverse orientation — and only its magnitude changes. The model can reweight how much each frozen direction matters, but cannot rotate any of them.
- **Direct (SPLIT-D).** $\widetilde{\mathbf{W}} = \mathbf{W}$: the basis is untouched and *entirely* determined by the initial orthogonalization, so all that training can do is choose coefficients $\mathbf{z}$ over fixed directions.

These are exact statements about what the parameterization can express. Only two of the three carry over to the detector's output: because the additive axis is a null direction, SPLIT is functionally a plain projection with identity activation, and its extra modulation buys no functional freedom. What actually separates the three configurations is therefore the multiplicative term (SPLIT-M versus SPLIT) and the output nonlinearity (SPLIT-D's $\tanh$ versus the identity in the other two), and any account of why they win on different datasets has to be an account of those two things rather than of the modulation alone. A magnitude-only rescaling of frozen directions reproduces a reweighting that the code already provides, so the natural expectation is that the multiplicative term and the nonlinearity matter mainly where the frozen directions are least adequate on their own—which is the regime we would expect for the more heterogeneous backbone. We state this as a conjecture about the dataset dependence in Table 11 rather than as a result; testing it would require training the missing cross-configurations (e.g. an additive generator with a nonlinear activation, or a direct projection with the identity) and we do not run those here.

Table 11: Win counts per (dataset, k) for the direct (D), additive (A) and additive-multiplicative (M) generators, out of the twelve image/pixel × CLIP/DINOv3 × metric cells of that (dataset, k). Each row of the table sums to twelve, so fractions such as 6.5 are split cells. The k=1 block is omitted for space.

	k=0			k=2		
	D	A	M	D	A	M
MVTec	3.0	2.5	6.5	1.0	2.0	9.0
VisA	7.5	2.5	2.0	5.5	3.5	3.0
BTAD	0.0	1.0	11.0	1.0	3.0	8.0
KSDD	11.5	0.0	0.5	11.3	0.3	0.3
Real-IAD	3.0	7.5	1.5	3.3	6.3	2.3

A.6 Deployment and inference cost

At equal *trainable* parameters the per-layer structure is cheaper structurally: a shared code needs a projection spanning the whole head ($L \cdot 2D \times d_1$) whereas a per-layer code re-uses one $2D \times d_2$ projection across layers, so $d_1 = Ld_2$ gives a frozen-basis ratio of $L^2 = 16$ at L=4. Under our implementation, which stores the frozen projection explicitly, the artifact inherits it (4.0 MB against 64.0 MB at d_1=2048) independently of the backbone, while throughput is unaffected—inference peak memory is identical for both and independent of d, being dominated by the frozen backbone (Tables 12, 13).

A second, optional saving is available *after* training: pruning a converged code reveals removable inference-time redundancy—nearly free on DINOv3 and improving CLIP—which we report as an orthogonal observation in Sec. A.7.

Full deployment-cost table (equal trainable budget). Every matched pair $d_1 = 4d_2$ for both backbones, with all four resource measures. Values are the actual saved checkpoint files, peak resident detector memory during inference, and per-image latency measured one structure per run on a single category (83 images), so no two entries contend for the GPU. `infer mem` is the peak during inference; it is identical between the two structures (1.09 GB on DINOv3, 0.78 GB on CLIP) and independent of d, so it is listed once per backbone.

Inference cost against prior methods. Table 13 places SPLIT in the same measurement harness as the baselines, on MVTec at batch size 8 and `num_workers`= 0. Two observations mirror the ones made from Table 12. First, the *model* footprint is set almost entirely by the backbone, not by the detector: every CLIP row sits at 1.59–1.64 GB and every DINOv3 row at 1.14–1.15 GB,

Table 12: Resource cost of a shared versus a per-layer code at equal trainable budget, for both backbones. The checkpoint ratio is $L^2 = 16$ in every row and is backbone-independent. Latency carries no structure-dependent signal (means: DINOv3 64.1 vs. 57.6 ms, CLIP 70.0 vs. 71.0 ms).

budget		checkpoint		model mem (GB)		latency (ms)	
d_1	d_2	shared	per-layer	shared	per-layer	shared	per-layer
DINOv3 (infer mem: 1.09 GB for both structures, all rows)							
64	16	2.0 MB	0.1 MB	1.15	1.14	73	57
128	32	4.0 MB	0.3 MB	1.15	1.14	62	57
256	64	8.0 MB	0.5 MB	1.16	1.15	57	57
512	128	16.0 MB	1.0 MB	1.17	1.14	59	56
1024	256	32.0 MB	2.0 MB	1.20	1.15	63	63
2048	512	64.0 MB	4.0 MB	1.27	1.15	74	56
3072	768	96.0 MB	6.0 MB	1.33	1.15	62	56
4096	1024	128.0 MB	8.0 MB	1.39	1.16	63	59
CLIP (infer mem: 0.78 GB for both structures, all rows)							
64	16	2.0 MB	0.1 MB	1.61	1.61	69	68
128	32	4.0 MB	0.3 MB	1.61	1.61	73	74
256	64	8.0 MB	0.5 MB	1.62	1.61	69	68
512	128	16.0 MB	1.0 MB	1.64	1.61	69	68
1024	256	32.0 MB	2.0 MB	1.66	1.61	73	68
2048	512	64.0 MB	4.0 MB	1.73	1.62	69	68
3072	768	96.0 MB	6.0 MB	1.79	1.62	69	80
4096	1024	128.0 MB	8.0 MB	1.86	1.61	69	74

independent of how the head is built. Second, SPLIT's inference peak memory (0.78 GB on CLIP, 1.09 GB on DINOv3) and per-image latency (71.0 ms on CLIP, 57.6 ms on DINOv3) stay within the range of the lightweight baselines while keeping the full multi-scale head—the mapping network is not what the runtime pays for. AnomalyCLIP is an outlier by more than an order of magnitude (894 ms/img), since it re-encodes text prompts and multi-scale windows at every query.

Table 13: Inference efficiency against prior methods on MVTec (batch size 8, `num_workers`= 0). *Model mem* is the resident detector footprint (backbone included); *infer. mem* is the peak allocation during a zero-shot (k=0) pass beyond that footprint; *latency* is per image, averaged over the test set. SPLIT entries are the per-layer measurements of Table 12 at its default latent dimension (d=2048 for CLIP, d=512 for DINOv3), with latency averaged over that table's eight budget points.

	AnomalyCLIP	**VisualAD**	**UniADet**		**LUMIN**		**SPLIT**	
	CLIP	CLIP	CLIP	DINOv3	CLIP	DINOv3	CLIP	DINOv3
Trainable params	5.52M	31.5M	0.02M	0.02M	0.002M	0.002M	0.008M	0.002M
Model mem (GB)	1.64	1.59	1.60	1.14	1.60	1.14	1.61	1.15
Infer. mem (GB)	3.20	2.00	0.83	1.09	0.78	1.09	0.78	1.09
Latency (ms/img)	894	115	88	74	88	75	71.0	57.6

Per-cell detail behind the cost comparison. *Inference peak memory* (the peak CUDA allocation during evaluation, reported as `infer_mem_GB`) is identical for the two structures (1.09 GB on DINOv3, 0.78 GB on CLIP) and independent of the latent dimension, because it is dominated by the frozen backbone and its activations: the detector's structure is not the bottleneck. *Latency* likewise shows no systematic difference: the means over all eight budget points are 64.1 vs. 57.6 ms on DINOv3 (per-layer faster in 7 of 8 pairs, but the shared timings themselves span 57–74 ms, so the 11% sits inside their own spread) and 70.0 vs. 71.0 ms on CLIP (1.11× and 0.99×). This is expected: a shared code performs $L \cdot 2D \cdot d_1$ multiply–adds against $2D \cdot d_2 \cdot L$ for the per-layer code—four times more at L=4—but both are negligible beside the backbone forward pass. The generated basis itself differs by the same factor: 64.0 MB against 4.0 MB at d_1=2048 and 16.0 MB against 1.0 MB at d_1=512, with resident model memory following (1.27 vs. 1.15 GB), and the ratio is unchanged in every row of Table 12.

A.7 LATENT TRUNCATION (TOP-K)

Setting budget through d changes the number of trainable parameters. A different constraint is to keep d trainable parameters but restrict how many of them are active in the forward pass. Hard Top-K truncation does this by keeping only the $K = \alpha d$ largest-magnitude entries of the latent vector and zeroing the rest at every forward pass (α=1 recovers the untruncated model). Because the operation sits inside the mapping network, it is active during *training* as well as inference: the model is trained under the constraint and is free to concentrate its latent onto the retained dimensions. We explored it on the per-layer latent (SPLIT, DINOv3, k=0). It yields only a marginal and unreliable gain. The positive region is confined to large d with aggressive truncation ($+2.82$ at d=2048 and $+3.69$ at d=4096, both with α=0.25), and neither is significant (p=0.208 and 0.088, paired t-test over the 15 categories; Table 14); both are moreover driven by a single category—at d=4096, only $8/15$ categories improve, with hazelnut gaining $+27.1$ while tile loses -5.6. The one significant positive cell is $+0.76$ (d=2048, α=0.75), negligible beside the > 8 points that d itself spans between 512 and 4096. In the opposite direction, truncation is significantly *harmful* wherever the retained width is small (-4.8 to -18.3 for $d \leq 512$ at α=0.25), consistent with the harm depending on the retained width $K = \alpha d$ rather than on α itself. Against this, truncation adds a hyper-parameter with no safe default—its optimum drifts with d (0.75 for $d \leq 256$, 0.5 for 512–3072, 0.25 for 2048–4096) and would require a joint search—and introduces a non-differentiable index-selection step into the forward pass. We therefore drop it and keep d as the budget knob.

Table 14: Training-time Top-K truncation. Change in MVTec pixel-AUPR (per-layer latent, DINOv3, k=0) with respect to the untruncated baseline (α=1) at the same d. The budget $K = \alpha d$ is fixed before training. The retained width—not the ratio α—determines the outcome: $d \leq 512$ at α=0.25 loses 4.8–18.3 points, whereas the largest d gains modestly. Only the positive cells are large-d/aggressive-α combinations, and both are insignificant and driven by a single category. *: paired t-test over the 15 categories, $p < 0.05$.

d	α=0.25	α=0.5	α=0.75
64	-18.32^*	-7.34^*	$+0.13$
128	-11.10^*	-4.57^*	-1.87^*
256	-6.31^*	-2.30	-0.46
512	-4.77^*	-0.26	-0.91
768	-0.24	$+0.98$	-0.54
1024	-1.21	$+0.09$	-0.03
1280	-0.05	-0.29	-0.14
1536	-1.10	$+0.41$	$+0.27$
2048	$+2.82$	$+0.45$	$+0.76^*$
3072	-0.71	$+0.61$	$+0.26$
4096	$+3.69$	$+0.35$	$+0.20$

The cost of truncation is an optimization cost, not a budget cost. Table 14 imposes the budget *during* training. To separate the two, we also truncate an already-trained α=1 code at inference only, with no retraining: the retained set is chosen once from the converged latent, so the model never has to allocate information under a shifting budget. Table 17 contrasts the two regimes at matched (d, α). The result is starkly different. Truncating a trained code is nearly free—at d=64, α=0.25 it costs 0.53 points, against 18.32 points when the same budget (K=16) is imposed during training; at d=128, α=0.25 the two are $+1.37$ and -11.10. Across all post-hoc cells the change lies in $[-4.5, +1.5]$ with only 5 of 39 significant, and the mean shift is -1.16 / $+0.03$ / -0.05 for α=0.25/0.5/0.75—i.e., the converged latent parameterization contains substantial removable coordinate redundancy, and dropping the smallest-magnitude dimensions discards information the optimizer had already left unused. At the best width we evaluate for DINOv3 (d=512) the same holds: truncating to half the dimensions costs nothing ($+0.66$, n.s.) and even truncating to a quarter costs only -3.57 (n.s.). The large penalty of train-time truncation must therefore come from the constraint itself, and it is concentrated where the budget binds hardest: because the retained set is recomputed from $\mathbf{z}$ at every step, a dimension can be active in one step and dead in the next, and the model cannot stably commit information to any fixed subset. That is fatal when the model must rely on every dimension (d=64, K=16), and harmless once K is large enough or d is already at its optimum

(d=512: -3.57 post-hoc vs. -4.77 trained). This is why we keep d, which fixes the budget before training begins, rather than a dynamic sparsification.

What post-hoc truncation buys in storage. A post-hoc truncation has one property the trained variant lacks: its retained set is fixed once, so the coordinates outside it are permanently zero and the columns of the frozen basis they multiply are never used. Those columns can therefore be pruned, and since the basis is the bulk of the artifact, the checkpoint scales with the retained width—≈2.0 MB at α=0.5 and ≈1.0 MB at α=0.25, against 4.0 MB untruncated, for the DINOv3 configuration of Table 12. The saving is available to an already-trained model without retraining it, which is the practical reason we report this study despite its null training-time result; it composes with the L^2 advantage of per-layer over shared codes, multiplying it by a further $1/\alpha$: at α=0.25 the parameter-matched pair of Table 12 spans 64× (1.0 MB against 64.0 MB at total budget 2048).

On CLIP the same post-hoc operation does not merely avoid harm—it helps. There, truncating a converged per-layer code to a quarter of its dimensions raises pixel-AUPR by $+8.90$ at d=128 and $+2.89$ at d=1024 (Table 16), and the gain is positive in $36/39$ cells and significant in 34 of them (mean $+2.95$ / $+1.97$ / $+0.25$ for α=0.25/0.5/0.75); it is broad-based rather than driven by outliers (at d=128, $13/15$ categories improve, median $+5.41$). CLIP therefore carries removable redundancy at inference even though its dimension curve is still rising at training time—budget it needs to *learn* the heterogeneous per-layer heads, but not to *apply* them.

The CLIP/DINOv3 sign reversal is not specific to MVTec: re-running the same post-hoc sweep with the same checkpoints on BTAD and KSDD (inference only, no retraining) reproduces it on all three evaluation sets (Table 15). At α=0.5 every CLIP cell improves on MVTec and BTAD ($48/48$), whereas every DINOv3 cell is non-positive on all three sets. The gain also holds for both latent structures, so it is not an artifact of the per-layer code. Two caveats remain: KSDD contains a single category and its cells are correspondingly noisy (direction preserved, magnitude smaller), and we still have no trained-truncation counterpart for CLIP, so the two regimes cannot be contrasted there. We therefore report this as an observation, not as a method; whether it survives other seeds, $k > 0$ and other modulations is untested.

Table 15: Cross-dataset replication of the post-hoc truncation effect, at α=0.5 (K=$d/2$). Mean change in pixel-AUPR relative to the untruncated baseline at the same d, with the number of d cells that improve in parentheses. The CLIP/DINOv3 sign reversal holds for both latent structures on all three evaluation sets, and every CLIP cell improves on MVTec and BTAD. KSDD has a single category, so its cells carry no cross-category averaging. DINOv3 rows are the α=0.5 columns of Table 17.

Backbone	Latent	MVTec	BTAD	KSDD
CLIP	per-layer	+1.97 (13/13)	+1.48 (13/13)	+1.38 (9/13)
CLIP	shared	+2.73 (11/11)	+2.53 (11/11)	+1.17 (8/11)
DINOv3	per-layer	+0.02 (5/13)	−1.08 (3/13)	−0.46 (7/13)
DINOv3	shared	−0.20 (3/11)	−0.17 (5/11)	−0.74 (4/11)

A.8 Feature consistency and visualizations

The quantitative CKA evidence is Figure 4 in the main text (Sec. 4.5); Figure 5 below shows the same contrast in the patch-feature geometry and is kept as supporting appendix material.

On DINOv3 (left) the four layers form a single, heavily interleaved manifold—the orange, green and red point clouds overlap and interpenetrate, with only the shallowest layer (blue) partially separated—indicating that adjacent-depth layers encode largely shared information. On CLIP (right) the four layers are cleanly segregated into four nearly parallel bands, ordered monotonically by depth, with almost no overlap between bands. The contrast visualizes the inter-layer heterogeneity of CLIP's contrastive features against the consistency of DINOv3's self-supervised features, corroborating the CKA statistics (mean cross-layer 0.845 for DINOv3 vs. 0.477 for CLIP). The visualization is used only as a qualitative companion to the CKA measurement; feature-depth utility is evaluated separately by the controlled window ablation in Sec. 4.6.

Table 16: Post-hoc truncation on CLIP. Change in MVTec pixel-AUPR (per-layer latent, CLIP ViT-L/14@336px, k=0) relative to the untruncated baseline (α=1) at the same d; absolute baselines as in the CLIP k=0 column of the additive dimension sweep above. Unlike DINOv3 (Table 17), truncating a converged code *improves* accuracy: 36 of 39 cells are positive and 34 are significant, including every cell with $d \geq 512$. The effect is broad-based rather than outlier-driven—at d=128, 13/15 categories improve with a median gain of $+5.41$. *: paired t-test over the 15 categories, $p < 0.05$.

d	α=0.25	α=0.5	α=0.75
16	-2.18	$+2.53^*$	-1.38^*
32	$+7.40^*$	$+1.42^*$	-0.28
64	$+1.37$	$+6.69^*$	$+2.17^*$
128	$+8.90^*$	$+5.24^*$	$+1.26^*$
256	$+3.71^*$	$+1.32^*$	$+0.15$
512	$+2.50^*$	$+1.20^*$	$+0.21^*$
768	$+2.75^*$	$+0.73^*$	$+0.13^*$
1024	$+2.89^*$	$+0.89^*$	$+0.16^*$
1280	$+2.82^*$	$+1.19^*$	$+0.18^*$
1536	$+3.02^*$	$+1.23^*$	$+0.10^*$
2048	$+2.26^*$	$+1.09^*$	$+0.30^*$
3072	$+1.07$	$+1.00^*$	$+0.10^*$
4096	$+1.80^*$	$+1.03^*$	$+0.19^*$

Table 17: Training-time versus post-hoc truncation at matched (d, α). Change in MVTec pixel-AUPR (per-layer latent, DINOv3, k=0) relative to the untruncated baseline at the same d. "Trained" imposes the budget $K = \alpha d$ during training (same data as Table 14); "post-hoc" truncates the converged α=1 code at inference and never retrains. The two diverge by up to 17.8 points where K is small—the same budget is nearly free when applied once to a converged code—and converge once K is large enough to be non-binding ($|\Delta| \leq 1.9$ at α=0.75 under both regimes).

	α=0.25		α=0.5		α=0.75	
d	trained	post-hoc	trained	post-hoc	trained	post-hoc
64	-18.32	-0.53	-7.34	$+0.37$	$+0.13$	-0.04
128	-11.10	$+1.37$	-4.57	$+0.89$	-1.87	-0.08
256	-6.31	-2.19	-2.30	-0.67	-0.46	-0.17
512	-4.77	-3.57	-0.26	$+0.66$	-0.91	-0.24
768	-0.24	-0.20	$+0.98$	-1.15	-0.54	$+0.05$
1024	-1.21	-4.48	$+0.09$	-1.00	-0.03	-0.09
1536	-1.10	-2.78	$+0.41$	-0.20	$+0.27$	$+0.06$
2048	$+2.82$	-1.69	$+0.45$	$+0.61$	$+0.76$	$+0.21$
4096	$+3.69$	-0.72	$+0.35$	-0.08	$+0.20$	$+0.05$

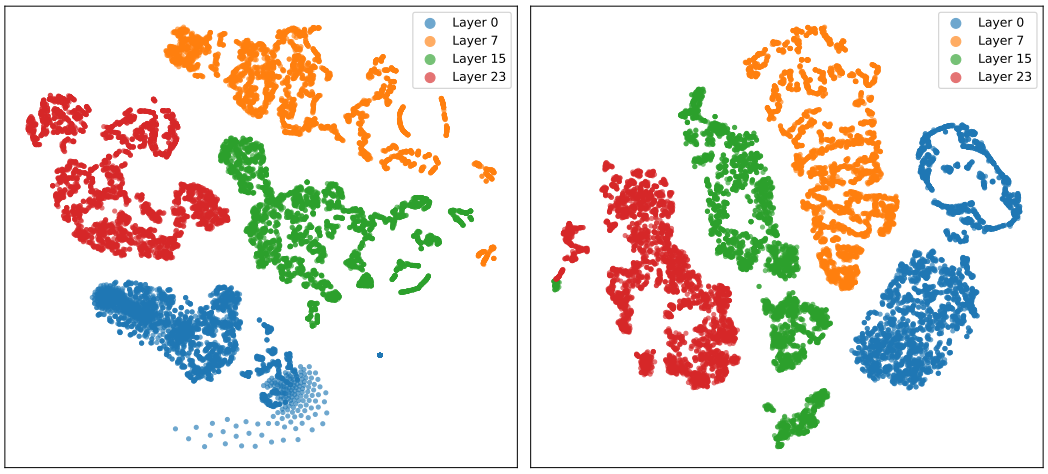

Figure 5: T-SNE (Van der Maaten & Hinton, 2008) for DINOv3 (left) and CLIP (right). Four layers $0, 7, 15, 23$ spanning shallow to deep are drawn in blue, orange, green and red respectively; both backbones use identical colors and the same patch budget (200 patches per layer per image $\times$ 12 images).

A.9 Full few-shot comparison

Table 18 gives the few-shot (k=1, 2) comparison summarized in Sec. 4: image- and pixel-level AUROC, AUPR and F1-max for UniADet, LUMIN and SPLIT on both backbones over all five benchmarks. Two patterns hold in every dataset block. First, the gain from k=0 to k=1 dominates the gain from k=1 to k=2—a single normal sample already supplies most of what a memory bank can contribute. Second, the ranking established zero-shot largely persists as shots are added, with one exception: LUMIN (CLIP), whose zero-shot pixel-AUPR collapses (12.2 on MVTec, 2.7 on KSDD) but recovers to competitive values from k=1 onward (48.7 and 39.3), i.e. a memory bank can compensate for the information lost by its single-head compression.

Table 18: Few-shot comparison against UniADet and LUMIN on both backbones for $k = 1, 2$. Each cell reports image-level / pixel-level AUROC, AUPR, F1-max. As in Table 1, all SPLIT cells use the same *fixed* configuration (additive generator, d=512/2048, window $[5, 11, 17, 23]$) with no per-dataset selection. **Red** = best value in each row segment (image-level or pixel-level), **blue** = second best.

k	Dataset	Metric	UniADet		LUMIN		SPLIT	
			CLIP	DINOv3	CLIP	DINOv3	CLIP	DINOv3
1			93.2/94.8	**93.5**/**96.6**	91.9/92.0	93.1/96.2	91.1/94.5	**93.9**/**96.5**
2		AUROC	93.4/95.2	93.8/**96.9**	**95.0**/92.3	93.3/96.5	91.7/94.8	**94.2**/**96.8**
1	MVTec		**96.9**/52.1	**97.1**/56.9	95.8/48.7	96.3/**57.1**	95.6/50.3	96.7/**58.7**
2		AUPR	**97.1**/53.5	**97.3**/**57.6**	97.0/52.0	96.4/**57.6**	96.1/51.8	96.8/**59.4**
1			92.8/53.0	93.0/**56.6**	**94.0**/52.0	92.3/55.6	91.5/52.0	**93.1**/**56.2**
2		F1max	92.9/54.5	**93.4**/**57.4**	**94.7**/54.3	92.4/55.9	91.9/53.6	93.3/**56.8**
1			**93.0**/96.5	**96.2**/**97.3**	85.6/96.2	92.3/**97.3**	91.2/96.6	92.4/**97.8**
2		AUROC	**93.5**/96.6	**96.4**/**97.4**	88.8/96.4	92.6/**97.4**	91.9/96.7	92.6/**97.8**
1	VisA		**75.5**/26.6	**86.2**/**30.2**	54.0/24.1	74.4/27.1	70.0/22.5	75.1/**29.6**
2		AUPR	**77.2**/27.5	**86.9**/**30.6**	60.5/26.6	75.5/27.6	71.9/23.2	76.3/**30.1**
1			71.2/33.3	**82.4**/**36.1**	56.7/32.3	72.1/33.3	67.3/28.7	**72.5**/**35.4**
2		F1max	73.1/34.3	**83.4**/**36.4**	61.7/34.7	73.1/33.8	69.5/29.7	**73.8**/**35.9**
1			93.5/95.7	**94.7**/**98.2**	92.0/90.1	**95.5**/**97.4**	91.0/95.7	92.7/97.0
2		AUROC	93.9/96.0	**94.8**/**98.2**	93.9/90.5	**95.6**/**97.4**	92.1/96.1	93.3/97.0
1	BTAD		**96.4**/54.1	**96.4**/**61.2**	88.3/46.2	**95.2**/**58.6**	92.6/52.0	89.2/54.0
2		AUPR	**96.8**/56.5	**96.2**/**61.5**	95.2/50.0	95.4/**59.0**	93.4/54.1	89.6/54.4
1			92.5/56.1	**94.6**/**57.9**	86.5/50.1	**93.5**/**56.2**	89.4/55.1	85.1/52.4
2		F1max	93.3/**57.6**	**94.6**/**58.2**	92.3/52.6	**93.5**/56.4	90.7/56.5	85.5/52.7
1			**96.0**/**97.2**	86.0/**98.8**	**98.2**/95.8	85.0/**98.8**	92.6/**97.2**	83.2/**98.8**
2		AUROC	**96.3**/97.2	86.2/**98.7**	**97.2**/95.9	84.8/**98.7**	93.4/97.2	83.4/**98.8**
1	KSDD		**77.1**/**45.0**	48.2/15.6	**89.8**/**39.3**	44.3/20.6	65.4/30.4	49.2/18.9
2		AUPR	**78.6**/**46.7**	48.3/15.6	**89.4**/**42.6**	43.8/20.6	69.8/31.7	49.4/19.0
1			**77.7**/38.3	54.6/26.5	**81.3**/**48.4**	51.0/32.0	62.0/**42.9**	58.2/27.8
2		F1max	**77.5**/**52.6**	55.7/26.6	**83.5**/**50.9**	50.9/31.9	65.0/43.4	59.5/28.0
1			76.7/96.4	**79.3**/**97.2**	67.2/94.3	**78.5**/**97.2**	76.9/96.2	76.8/**97.1**
2		AUROC	77.7/96.8	**79.9**/**97.5**	70.6/94.9	**79.1**/**97.5**	77.9/96.6	77.4/**97.4**
1	Real-IAD		84.8/37.7	**87.3**/**43.1**	79.6/19.1	**87.0**/**41.9**	86.0/36.4	85.3/39.5
2		AUPR	85.4/38.8	**87.7**/**44.0**	80.9/21.2	**87.6**/**43.0**	86.6/37.5	85.7/40.5
1			82.2/41.7	**82.9**/**45.4**	80.2/25.6	**82.3**/**44.4**	82.0/40.9	81.7/42.1
2		F1max	**82.5**/42.3	**83.1**/**46.0**	81.1/27.1	82.4/**45.0**	82.3/41.5	81.9/42.7