DO HIERARCHICAL VISION BACKBONES NEED A FOURTH STAGE?

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical vision backbones devote substantial capacity to their final stage. We investigate how much of this capacity is needed and how learning without it changes the preceding hierarchy. We train Swin and ConvNeXt models from Nano to Base with and without the fourth-stage blocks, always keeping the final downsampling transition. Small and Base models trained without these blocks use about 29% fewer parameters and 25% less peak inference memory at batch size one, and lose only 0.13–0.53 ImageNet-1K Top-1 points. Reallocating this capacity upstream can improve accuracy at comparable parameter counts, at the cost of more FLOPs, whereas the narrowest configurations still favor four stages. Under a common normalized readout, class information becomes accessible earlier within the third stage in all ten model pairs, and truncated models generally depend more on the late spatial exchanges of this stage. On Swin-T-C64, a terminal token-mixing module with 8.6% of the original stage's parameters recovers about half of the truncation loss and outperforms a parameter-matched channel-only control. COCO box and mask AP stay within one point of the full models in the evaluated configurations, and ImageNet-C gaps are modest at Small and Base scale but larger in some smaller configurations. These findings support choosing terminal capacity jointly with upstream allocation and the evaluation objective.

1 INTRODUCTION

Most hierarchical vision backbones follow a four-stage design in which spatial resolution decreases and channel width increases (He et al., 2016; Liu et al., 2021; 2022b; Li et al., 2024; Liu et al., 2024b). The final stage S_4 is parameter-dense: in Swin and ConvNeXt Small and Base, its two or three blocks hold nearly 30% of the parameters while operating on a 7×7 grid. Competitive three-stage architectures such as HiViT (Zhang et al., 2023) indicate that this stage is not structurally required. We ask how much of the final stage a backbone needs, and how the preceding hierarchy organizes itself in its absence.

We study Swin and ConvNeXt models trained from scratch with and without S_4, from Nano to Base (Figure 1). Truncation removes the blocks of S_4 while retaining the final downsampling transition D_3 and the stride-32 feature map.

The utility of S_4 depends strongly on the budget. At Small and Base scale, models trained without it lose only 0.13–0.53 ImageNet-1K Top-1 points while using about 29% fewer parameters and 25% less peak memory at batch size one. Reallocating the removed capacity to S_3 can improve accuracy at comparable parameter counts, albeit at higher FLOPs. In the narrowest configurations, however, removal is much costlier, and a four-stage allocation remains preferable.

Learning without S_4 also changes the preceding hierarchy. Under a common normalized readout protocol, class information becomes accessible at shallower depths within S_3 in all ten pairs, with the largest differences inside S_3 rather than at its output. Full and truncated models have similar pooled representations at the second downsampling transition but diverge from S_3 onward. Truncated models also generally depend more on the spatial exchanges of their last S_3 blocks.

Terminal computation can nevertheless remain useful at a fraction of the original capacity. On Swin-T-C64, a two-block terminal token-mixing module with 8.6% of S_4's parameters recovers about half of the truncation loss, whereas a parameter-matched channel-only module recovers none.

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.

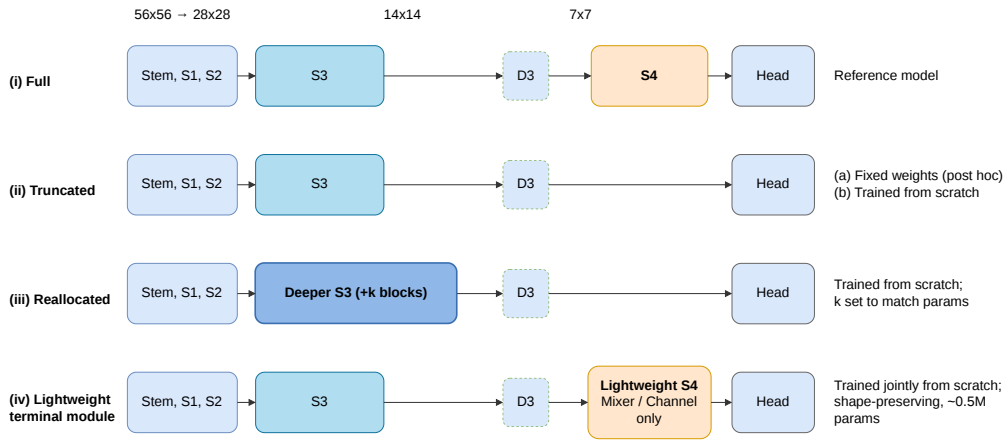


Figure 1: **Four interventions on the fourth stage.** Removing S_4 at fixed weights measures reliance; training the truncated model from scratch measures necessity; deepening S_3 tests capacity allocation; and lightweight terminal modules test terminal computation. All variants retain the final downsampling transition D_3 and the stride-32 feature map.

COCO (Lin et al., 2014) box and mask AP stay within one point of the full models in the evaluated Tiny-C64 and Small configurations. On ImageNet-C (Hendrycks & Dietterich, 2019), accuracy gaps exceed those on clean images but remain modest at Small and Base scale, with larger losses in some smaller configurations. The number of stages should therefore be chosen jointly with upstream capacity and the evaluation objective rather than fixed by convention.

Our contributions are as follows:

1. **Budget-dependent utility of S_4.** We characterize when removing or reallocating terminal capacity preserves accuracy and quantify the resulting parameter and memory savings (Section 4.1).
2. **Reorganization of the preceding hierarchy.** Without S_4, class information becomes accessible earlier within S_3, and models generally rely more on its late spatial exchanges (Section 4.2).
3. **Lightweight terminal computation.** On Swin-T-C64, a terminal module using under 10% of S_4's parameters recovers about half the truncation loss and outperforms a parameter-matched channel-only control (Section 4.3).

2 Related Work

2.1 Staged Hierarchical Architectures

Modern hierarchical backbones commonly use four stages, lowering spatial resolution and raising channel dimensionality as depth increases (Wang et al., 2021; Li et al., 2022; Shaker et al., 2023; Pan et al., 2022).

Three-stage alternatives include HiViT, iTPN, Fast-iTPN, and LeViT (Zhang et al., 2023; Tian et al., 2023; 2024; Graham et al., 2021; Wu et al., 2026). In HiViT's transition from ViT to Swin, adding a fourth stage yielded only marginal gains, which motivated an architecture without S_4 (Zhang et al., 2023). This comparison does not isolate the fourth-stage blocks: adding S_4 moves two blocks from S_3 to S_4, replaces global attention in S_3 with shifted-window attention, and introduces the final $14 \times 14 \to 7 \times 7$ transition. It thus changes capacity allocation, spatial mixing, and terminal resolution at once, since the three-stage model ends at the S_3 resolution.

We instead remove only the S_4 blocks and retain the preceding hierarchy, D_3, and the stride-32 feature map. Across five configurations per family, from Nano to Base, we study fixed-weight

removal and learning without S_4, with separate controls for capacity reallocation and lightweight terminal modules. These interventions separate reliance on the terminal stack from adaptation in S_3 and D_3, from capacity allocation, and from terminal computation. Three-stage designs show that four stages are not mandatory; we ask how much of an existing backbone can be removed without losing accuracy.

2.2 Capacity allocation, structural compression, and reallocation

EfficientNet studies the joint scaling of depth, width, and resolution (Tan & Le, 2019), while RegNet explores design spaces with varying stage widths and depths (Radosavovic et al., 2020). Work on stage-level allocation further emphasizes where computation is spent (Liu et al., 2022a). ConvNeXt offers a concrete example: changing the ResNet-50 (He et al., 2016) block allocation from $(3, 4, 6, 3)$ to $(3, 3, 9, 3)$, following Swin-T, improves ImageNet accuracy (Liu et al., 2022b; He et al., 2016; Liu et al., 2021).

Structured pruning raises a related distinction between inherited weights and the architecture they leave behind. Liu et al. (2019) find that training several structured-pruned architectures from random initialization matches or exceeds fine-tuning the inherited weights. Depth-pruning methods such as UPDP and HetDPT study block removal and the functional heterogeneity of layers (Liu et al., 2024a; Su et al., 2026).

These studies show that the distribution of depth, width, and computation across stages matters, but they do not directly test whether the capacity of S_4 is tied to its terminal position, or whether it can be removed or spent in S_3 while the final downsampling transition is preserved. Comparing fixed-weight removal with independent training separates reliance on pretrained representations from what the architecture can learn. Reallocation tests where capacity is placed at comparable parameter counts, at the cost of more computation on the larger grid of S_3. Our analysis bears on structured pruning but is not itself a compression method.

2.3 Intermediate representations and adaptation

Linear classifier probes measure the task information accessible from frozen intermediate features (Alain & Bengio, 2017). CKA compares learned representations (Kornblith et al., 2019), and layer-wise similarity analyses trace how they progress within a model (Jiang et al., 2025). Comparisons of CNNs and vision transformers reveal architectural differences in that progression and in spatial integration (Raghu et al., 2021; Park & Kim, 2022). Concept analyses describe semantic complexity growing with depth (Dorszewski et al., 2026), and ERF measurements characterize spatial input sensitivity (Luo et al., 2016; Shi, 2024).

Within a fixed network, linear separability can saturate before the last layers, which then mainly compress representations (Masarczyk et al., 2023). Masarczyk et al. (2023) also train shallower networks from scratch on CIFAR-10 tasks but report only their accuracy, without comparing the resulting representations. These works mainly characterize representations within a given hierarchy. We instead ask whether intermediate representations depend on the stages that follow them. We compare matching S_3 and D_3 outputs of independently trained full and truncated networks with block-wise controlled heads, CKA, and ERFs, which tests reorganization when S_4 is absent during learning, not removed after training.

3 Method

3.1 Terminal stage and capacity allocation

Let E_θ denote the network from the stem through S_3, including the first two downsampling transitions. For an input image x, we write

$$z_3 = E_\theta(x), \qquad u = D_3(z_3), \qquad f(x) = H_\phi(T_\psi(u)), \tag{1}$$

where H_ϕ is the classification head and, for a 224×224 input to Swin or ConvNeXt (Liu et al., 2021; 2022b), D_3 maps the 14×14 grid to 7×7 and doubles the channel count. Full models use $T_\psi = S_{4,\psi}$, whereas truncated models set T_ψ to the identity; both retain D_3, the stride-32 feature

map, and the native head architecture. A lightweight replacement instead uses a learned T_ψ with the same input and output shapes.

Full and truncated models are trained separately from random initialization, with matched recipes within each family. Allocation controls deepen S_3 or redistribute blocks between S_3 and S_4 at approximately matched parameter counts, with stage widths, D_3, and the head architecture unchanged. Because S_3 operates on a larger grid, matching parameters does not match FLOPs. Appendix A documents the exact architectures, the training recipes, and additional interventions on pretrained models.

3.2 Representation measurements

At each attachment point, we freeze the backbone and fit a new head

$$H_3(Z) = A(\mathrm{LN}(P(Z))), \tag{2}$$

where P is global average pooling, LN is LayerNorm (Ba et al., 2016), and A is an affine classifier. All families share this normalization protocol, but each head is fitted separately on the full ImageNet-1K training set and evaluated at its final epoch. Block-wise profiles cover D_2, every S_3 block, D_3, and every S_4 block of full models. Throughout, *earlier* refers to relative depth within S_3, not to training time.

For relative depth $\rho \in [0, 1]$, with D_2 at $\rho = 0$, let $\delta(\rho) = \mathrm{Acc}_{H_3}(T, \rho) - \mathrm{Acc}_{H_3}(F, \rho)$ for a comparator T and its full reference F. We summarize the change within S_3 by

$$\Delta_{\mathrm{prog}} = \int_0^1 [\delta(\rho) - \delta(0)]\, d\rho, \tag{3}$$

computed with linear interpolation and the trapezoidal rule. Subtracting the input gap $\delta(0)$ isolates the change that develops within the stage.

We also measure linear centered kernel alignment (CKA) on pooled features (Kornblith et al., 2019) and gradient-based effective receptive fields (ERFs) (Luo et al., 2016). The ERF radius r_{50} encloses half the mass within the largest probe-centered disk contained in the image. Image-level intervals are conditional on the evaluated checkpoints. Appendix A gives the head-training settings, alternative readouts, and measurement definitions.

3.3 Terminal modules and spatial restrictions

On Swin-T-C64, we replace S_4 with either a two-block MLP-Mixer (Tolstikhin et al., 2021) or a channel-only module of approximately the same parameter count. Both are trained jointly with the backbone to test the contribution of terminal spatial mixing.

To measure fixed-weight sensitivity to late-S_3 spatial exchanges, we remove the window shifts of Swin, or mask the 7×7 depthwise kernels of ConvNeXt to their central 3×3 taps, in the last $k \in \{2, 4, 8\}$ blocks of S_3. Appendix A gives the module definitions and training settings, details these interventions, and reports restrictions within S_4 and a short factorial training experiment.

4 Experiments

We evaluate ImageNet-1K classification at 224×224 resolution (Russakovsky et al., 2015). Full and truncated models are trained for 300 epochs with matched recipes within each family, and the stochastic-depth slope is preserved when block counts change. We report the best validation Top-1; results for our locally trained ConvNeXt models use non-EMA weights. Parameter counts include the classification head. Appendix A provides the full protocols, checkpoint-level prediction tests, and analyses of pretrained models.

4.1 Utility of the final stage across budgets

Table 1 reports all ten pairs. For Swin and ConvNeXt Small/Base, truncation removes 28.5–28.7% of the parameters and about 8% of the FLOPs at a cost of 0.13–0.53 Top-1 points. Peak allocated

Table 1: Full and truncated models trained from scratch. Differences are truncated minus full; accuracies are truncated to two decimals after computing differences. Memory is peak allocated GPU memory in FP32 at batch size one. Absolute memory, FLOPs, and throughput are in Appendix A.

Model	Params (M) Full / Trunc.	Removed (%)	Top-1 (%) Full / Trunc.	Δ Top-1 (pp)	Δ peak memory (%)
Swin-B	87.7 / 62.5	28.7	83.33 / 83.19	−0.13	−25.9
Swin-S	49.6 / 35.4	28.6	83.04 / 82.50	−0.53	−24.8
Swin-T	28.2 / 14.1	50.1	81.20 / 78.78	−2.42	−40.4
Swin-T-C64	22.2 / 15.9	28.4	81.18 / 80.39	−0.79	−23.0
Swin-Nano-C32	5.7 / 4.1	27.6	73.84 / 70.98	−2.86	−16.0
ConvNeXt-B	88.5 / 63.2	28.6	83.84 / 83.39	−0.44	−26.7
ConvNeXt-S	50.2 / 35.9	28.5	83.23 / 82.91	−0.31	−25.4
ConvNeXt-T	28.5 / 14.3	50.0	82.04 / 80.65	−1.38	−42.6
ConvNeXt-T-C64	22.6 / 16.2	28.2	81.69 / 81.32	−0.36	−24.3
ConvNeXt-Nano-C32	4.1 / 3.0	26.3	74.32 / 71.24	−3.08	−14.6

GPU memory at batch size one falls by 24.8–26.7% in these four pairs and by 14.6–42.6% across all ten.

Standard Tiny models lose 2.42 points for Swin and 1.38 for ConvNeXt, but truncation also removes roughly half of their parameters: their normalized stage-depth allocation is $[1, 1, 3, 1]$, versus $[1, 1, 9, 1]$ for Small/Base. The $C = 64$ controls keep the Small/Base allocation and lose 0.79 and 0.36 points with 28.4% and 28.2% fewer parameters. At $C = 32$, the Nano models of both families (depths $[2, 2, 18, 2]$) lose 2.86 and 3.08 points with 27.6% and 26.3% fewer parameters.

Capacity allocation. Deepening S_3 to approximately match the parameter budget of the full Tiny models yields 81.88 Top-1 for Swin and 82.22 for ConvNeXt, 0.67 and 0.18 points above their full baselines. At $C = 64$, three-stage allocations exceed parameter-matched four-stage controls by 0.41 and 0.30 points. At $C = 32$, the preference reverses: four-stage $[2, 2, 10, 2]$ controls exceed the truncated $[2, 2, 18]$ models by 0.85 and 1.08 points with slightly fewer parameters, and reallocating the full Nano budget to S_3 ($[2, 2, 26]$) still leaves gaps of 0.32 and 0.78 points to the full models. The reallocated variants require more FLOPs; Appendix A reports their exact budgets and architectures.

4.2 Representations within S_3 and D_3

Under the common H_3 protocol, truncated models reach a given readout accuracy at shallower relative depths within S_3 in all ten pairs (Figure 2a–d). Δ_{prog} is positive in every pair, from 1.55 to 5.79 points for Swin and from 1.39 to 5.74 for ConvNeXt, and its image-level 95% intervals, conditional on the checkpoints, exclude zero (Table S9). The gap peaks inside S_3 and narrows over the last blocks.

At Nano-C32, both tested allocations without S_4 exceed both allocations with S_4 by at least 14.5 readout points at the end of S_3 in each family (panels e,f). This ordering holds for S_3 depths from 10 to 26 blocks and within both parameter-matched pairs, and the parameter-matched reallocated Swin-T also has a positive Δ_{prog} (12.19 points). In full models, the readout gain from the end of S_3 to the final S_4 block falls from 16.76 and 18.75 points at Nano to -0.18 and -0.54 at Base (panel h); measured from D_3, the Base gains are 0.34 and 0.90 points. Appendix A reports all profiles and boundary readouts.

Similarity and receptive fields. CKA between full and truncated models ranges from 0.941 to 0.997 at D_2, from 0.163 to 0.736 at S_3, and from 0.078 to 0.520 at D_3 (Table 2). At S_3, r_{50} is larger for the truncated Small and Base pairs, both ConvNeXt Tiny pairs, and ConvNeXt-Nano, whereas it is slightly smaller for the two Swin Tiny pairs and Swin-Nano. Terminal-output ERFs, CKA at the other attachment points, block-by-block CKA, and pooled-feature geometry are reported in Appendix A.

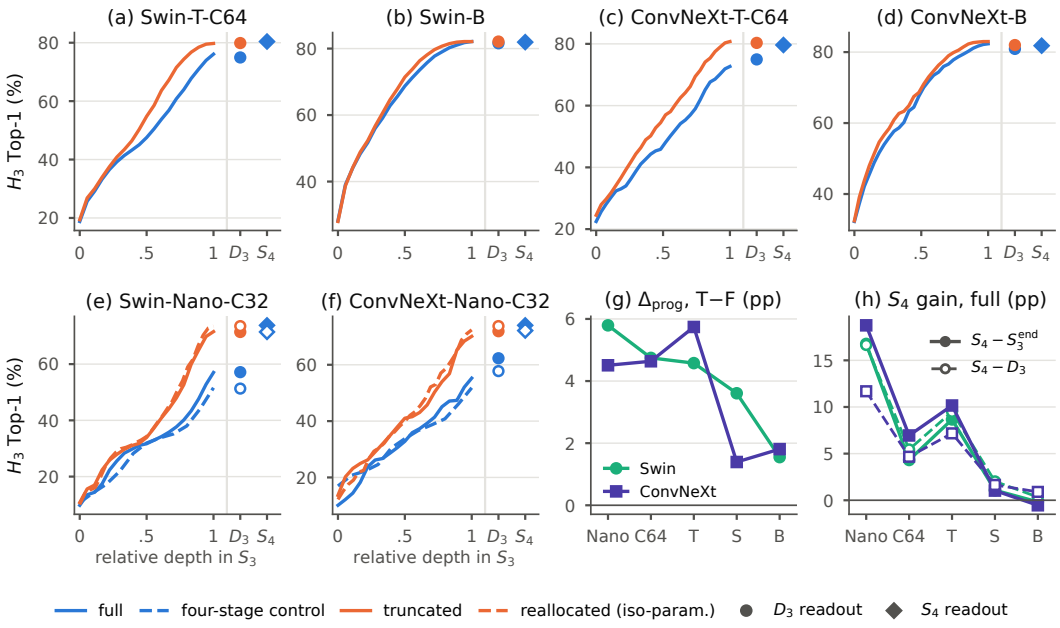


Figure 2: **Class decodability within S_3 under a common H_3 readout protocol.** (a–d) Top-1 of H_3 heads along relative depth in S_3 (D_2 at 0) for representative full and truncated pairs; markers give the D_3 and final-S_4 readouts. (e,f) Nano-C32 allocation controls with S_4 (full $[2, 2, 18, 2]$; four-stage $[2, 2, 10, 2]$) and without S_4 (truncated $[2, 2, 18]$; reallocated $[2, 2, 26]$). In (e,f), filled markers correspond to full/truncated models, and open markers to four-stage/reallocated models; circles denote D_3 readouts and diamonds denote final-S_4 readouts. (g) Δ_{prog} (Eq. 3) for the ten pairs; C64 denotes T-C64. (h) Accuracy added by S_4 in full models, relative to the end of S_3 (filled) and to D_3 (open). All profiles are in Appendix A.

Table 2: Representation comparison of full and truncated checkpoints. CKA measures pooled-feature similarity; S_3 ERF r_{50} is the radius, in input pixels, that encloses half the mass of the largest probe-centered disk contained in the image.

	Full/truncated CKA			S_3 ERF r_{50}	
Model	D_2	S_3	D_3	Full	Truncated
Swin-Nano-C32	0.993	0.163	0.236	55.032	54.208
Swin-T-C64	0.996	0.241	0.188	51.793	51.580
Swin-T	0.977	0.436	0.362	58.639	56.996
Swin-S	0.996	0.373	0.467	50.403	64.904
Swin-B	0.997	0.606	0.520	48.749	62.020
ConvNeXt-Nano-C32	0.941	0.485	0.106	45.349	55.682
ConvNeXt-T-C64	0.979	0.545	0.173	50.700	56.573
ConvNeXt-T	0.975	0.432	0.078	42.432	53.202
ConvNeXt-S	0.983	0.675	0.275	47.397	53.240
ConvNeXt-B	0.989	0.736	0.301	47.566	53.353

Spatial sensitivity. Restricting spatial exchanges in the last S_3 blocks at fixed weights lowers accuracy more in the truncated model than in its full counterpart in 27 of 29 conditions; the exceptions are ConvNeXt-Nano with four and eight restricted blocks. Appendix A reports all restriction results.

4.3 LIGHTWEIGHT TERMINAL REPLACEMENTS

Table 3 compares the two terminal replacements on Swin-T-C64. The Mixer reaches 80.78 Top-1, 0.51 points above the channel-only control (80.27), with parameter counts that differ by only 450. Relative to the slope-matched truncated reference (80.39), the Mixer gains 0.38 points with 0.540M additional parameters, about 8.6% of the parameters of the native S_4 blocks; the full model reaches 81.18. Appendix A gives the exact module widths and stochastic-depth rates.

Table 3: Terminal spatial-mixing control on Swin-T-C64. Best validation Top-1 after 300 epochs; Δ is relative to the full S_4 model, in percentage points.

Swin-T-C64 variant	Params (M)	FLOPs (G)	Top-1 (%)	Δ Top-1 (pp)
Full S_4	22.258	3.943	81.18	—
Truncated	15.948	3.629	80.39	-0.79
Two-block terminal Mixer	16.487	3.660	80.78	-0.40
Two-block channel-only control	16.488	3.656	80.27	-0.91

4.4 ROBUSTNESS AND DOWNSTREAM TRANSFER

Table 4 reports Top-1 accuracy on the three benchmarks. On ImageNet-C (Hendrycks & Dietterich, 2019), averaged over 15 corruptions and five severities, the full-minus-truncated Top-1 gap exceeds the clean gap, measured with the same weights in the same FP32 evaluation, in all ten pairs. It ranges from 0.56 to 1.21 points for Small/Base and reaches 5.95 for Swin-T. On ImageNet-v2 (Recht et al., 2019), full models lead in eight pairs, whereas the truncated ConvNeXt-T-C64 and ConvNeXt-B models gain 0.23 and 0.24 points. With the ReaL labels (Beyer et al., 2020), truncation costs less than with the original labels in all ten pairs. Appendix A reports the full results and intermediate readouts under corruption.

Table 4: Top-1 accuracy (%) of full and truncated models on ImageNet-C, ImageNet-v2, and ImageNet-ReaL.

	ImageNet-C		ImageNet-v2		ImageNet-ReaL	
Model	Full	Truncated	Full	Truncated	Full	Truncated
Swin-Nano-C32	41.81	38.04	61.65	58.88	81.28	78.83
Swin-T-C64	54.52	52.94	70.16	68.89	86.84	86.29
Swin-T	53.17	47.21	70.28	67.50	86.67	85.33
Swin-S	58.00	56.79	72.37	71.42	87.76	87.31
Swin-B	57.93	57.37	72.74	71.92	87.82	87.76
ConvNeXt-Nano-C32	43.91	40.08	62.17	59.50	82.04	79.47
ConvNeXt-T-C64	58.13	56.28	70.32	70.55	86.84	86.83
ConvNeXt-T	57.34	54.08	70.96	69.51	86.89	86.42
ConvNeXt-S	61.40	60.28	72.72	72.53	87.99	87.75
ConvNeXt-B	62.16	61.42	72.93	73.17	88.11	87.82

After 24 epochs of Mask R-CNN (He et al., 2017) training on COCO, truncation lowers box AP by 0.3–0.9 points and mask AP by 0.5–0.9 points in the Tiny-C64 and Small pairs (Table 5). Appendix A gives all six AP metrics and the training protocol.

Table 5: COCO `val2017` AP after 24 epochs of Mask R-CNN training. Δ is truncated minus full AP, in points.

	Box AP			Mask AP		
Model	Full	Trunc.	Δ	Full	Trunc.	Δ
Swin-S	47.7	46.9	-0.8	42.8	42.1	-0.7
ConvNeXt-S	46.9	46.0	-0.9	42.6	41.7	-0.9
Swin-T-C64	45.8	45.2	-0.6	41.2	40.7	-0.5
ConvNeXt-T-C64	45.6	45.3	-0.3	41.2	40.7	-0.5

5 Discussion

Our results tie the value of S_4 to upstream capacity, to the representations learned before it, and to terminal computation.

Capacity and its allocation. At Small and Base scale, a large part of the parameter budget can be removed at little cost in clean accuracy, and peak memory drops as well. Standard Tiny models lose more, but truncation also removes roughly half of their parameters, so width and stage allocation must be considered together. ConvNeXt-T-C64 loses 0.36 Top-1 points, close to the Small/Base losses, whereas Swin-T-C64 loses 0.79 points; this difference may partly reflect the deeper S_3 of ConvNeXt-T-C64 (27 versus 18 blocks). Reallocating capacity to S_3 improves Tiny accuracy at comparable parameter counts, at the cost of more FLOPs, and the allocation controls favor S_3 at $C = 64$ but four stages at $C = 32$.

Reorganization of the surviving hierarchy. Earlier decodability within S_3 in all ten pairs, lower CKA at S_3 and D_3 than at D_2 (Table 2), and the changes in spatial sensitivity are consistent with upstream representations adapting to the absence of S_4. The Nano allocation controls show the same pattern across the tested depths and budgets. The largest readout differences arise inside S_3, not at its output, which suggests that discriminative processing is redistributed through the stage. ERFs widen or narrow depending on family and scale, so the common shift in class accessibility coexists with different spatial adaptations.

Useful computation with less terminal capacity. Pre-pooling readouts on frozen Tiny features suggest a role for nonlinear processing at each position before aggregation. On Swin-T-C64, the Mixer recovers about half of the truncation loss with 8.6% of the parameters of S_4 and outperforms a parameter-matched channel-only control, which points to an additional contribution from terminal spatial mixing. Token-permutation controls in full Swin models suggest that terminal aggregation can be relatively insensitive to spatial arrangement. Both the generally greater sensitivity of truncated models to late-S_3 restrictions and the early-training Swin-Nano factorial (Appendix A) support a complementarity between upstream and terminal mixing. Whether the final truncated representation reduces the need for terminal mixing remains to be tested.

Transfer, robustness, and perspectives. The tested Tiny-C64 and Small models retain most of their COCO accuracy while saving backbone parameters, with box and mask AP losses below one point. ImageNet-C gaps exceed the clean losses but amount to 0.56–1.21 points at Small/Base (Table 4). The smaller ReaL losses and the two ImageNet-v2 gains also show that the effect of truncation depends on the evaluation setting. Terminal capacity should therefore be chosen jointly with resource constraints and the target evaluation conditions.

This evidence has a limited scope: training comparisons cover Swin and ConvNeXt, terminal replacements only Swin-T-C64, and COCO transfer only the Tiny-C64 and Small pairs. The short factorial covers two seeds during warmup, image-level intervals are conditional on the evaluated checkpoints and heads, and memory and throughput were measured on one GPU in FP32. Future work could trace the computations that link upstream reorganization to accuracy retention and test whether lightweight terminal modules keep their benefit under corruption. Appendix A provides an extended discussion and the complete protocols.

6 CONCLUSION

The value of the fourth stage depends on how capacity and computation are distributed across the hierarchy. In Swin and ConvNeXt, training without S_4 changes the organization of S_3 and lets Small and Base models retain nearly all of their clean accuracy with about 29% fewer parameters and substantially lower peak memory. The larger cost in narrow models shows the limits of this reduction.

On Swin-T-C64, a lightweight terminal mixer recovers part of the truncation loss, which shows that useful terminal computation can be delivered with a small fraction of the parameters of a conventional fourth stage. Given the measured transfer and robustness trade-offs, terminal capacity should be chosen jointly with upstream allocation and the evaluation objective.

AI USE STATEMENT

In this work, we used generative AI tools solely for English translation and language editing of the manuscript. The use of AI was limited to improving grammar, clarity, and readability.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or sensitive data.

REPRODUCIBILITY STATEMENT

The paper and Appendix A describe the experimental setup, model configurations, training procedures, and evaluation protocols in detail. All experiments use publicly available datasets, including ImageNet-1K.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2017. URL https://openreview.net/forum?id=ryF7rTqgl.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with ImageNet? *arXiv preprint arXiv:2006.07159*, 2020. URL https://arxiv.org/abs/2006.07159.

Teresa Dorszewski, Lenka Tětková, Robert Jenssen, Lars Kai Hansen, and Kristoffer Knutsen Wickstrøm. From colors to classes: Emergence of concepts in vision transformers. In Riccardo Guidotti, Ute Schmid, and Luca Longo (eds.), *Explainable Artificial Intelligence*, pp. 28–47, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-08317-3.

Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Herve Jegou, and Matthijs Douze. LeViT: a vision transformer in ConvNet's clothing for faster inference. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12239–12249. IEEE, October 2021. doi: 10.1109/iccv48922.2021.01204. URL http://dx.doi.org/10.1109/ICCV48922.2021.01204.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017. doi: 10.1109/ICCV.2017.322.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJz6tiCqYm.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs), 2016. URL https://arxiv.org/abs/1606.08415.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. *Deep Networks with Stochastic Depth*, pp. 646–661. Springer International Publishing, 2016. ISBN 9783319464930. doi: 10.1007/978-3-319-46493-0_39. URL http://dx.doi.org/10.1007/978-3-319-46493-0_39.

Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. Tracing representation progression: Analyzing and enhancing layer-wise similarity. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vVxeFSR4fU.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Siyuan Li, Zedong Wang, Zicheng Liu, Cheng Tan, Haitao Lin, Di Wu, Zhiyuan Chen, Jiangbin Zheng, and Stan Z Li. MogaNet: Multi-order gated aggregation network. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 37393–37427, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/a216c27f2f3160b1785c057fa510fdf1-Paper-Conference.pdf.

Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. EfficientFormer: Vision transformers at MobileNet speed. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 12934–12949. Curran Associates, Inc., 2022. doi: 10.52202/068431-0940. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/5452ad8ee6ea6e7dc41db1cbd31ba0b8-Paper-Conference.pdf.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network, 2013. URL https://arxiv.org/abs/1312.4400.

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*, pp. 740–755. Springer International Publishing, 2014. ISBN 9783319106021. doi: 10.1007/978-3-319-10602-1_48. URL http://dx.doi.org/10.1007/978-3-319-10602-1_48.

Chuanjian Liu, Kai Han, An Xiao, Ying Nie, Wei Zhang, and Yunhe Wang. Network amplification with efficient MACs allocation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1932–1941, 2022a. doi: 10.1109/CVPRW56347.2022.00210.

Ji Liu, Dehua Tang, Yuanxian Huang, Li Zhang, Xiaocheng Zeng, Dong Li, Mingjie Lu, Jinzhang Peng, Yu Wang, Fan Jiang, Lu Tian, and Ashish Sirasao. UPDP: a unified progressive depth pruner for CNN and vision transformer. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024a. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i12.29296. URL https://doi.org/10.1609/aaai.v38i12.29296.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. VMamba: Visual state space model. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 103031–103063. Curran Associates, Inc., 2024b. doi: 10.52202/

079017-3273. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/baa2da9ae4bfed26520bb61d259a3653-Paper-Conference.pdf.

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 *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021. doi: 10.1109/ICCV48922.2021.00986.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJlnB3C5Ym.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11966–11976, 2022b. doi: 10.1109/CVPR52688.2022.01167.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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*, volume 29, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/c8067ad1937f728f51288b3eb986afaa-Abstract.html.

Wojciech Masarczyk, Mateusz Ostaszewski, Ehsan Imani, Razvan Pascanu, Piotr Mił oś, and Tomasz Trzcinski. The tunnel effect: Building data representations in deep neural networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 76772–76805. Curran Associates, Inc., 2023. doi: 10.52202/075280-3355. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/f249db9ab5975586f36df46f8958c008-Paper-Conference.pdf.

Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. EdgeViTs: Competing light-weight CNNs on mobile devices with vision transformers. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pp. 294–311, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20082-3. doi: 10.1007/978-3-031-20083-0_18. URL https://doi.org/10.1007/978-3-031-20083-0_18.

Namuk Park and Songkuk Kim. How do vision transformers work? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=D78Go4hVcxO.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10425–10433, 2020. doi: 10.1109/CVPR42600.2020.01044.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 12116–12128. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/652cf38361a209088302ba2b8b7f51e0-Paper.pdf.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5389–5400. PMLR, 2019. URL https://proceedings.mlr.press/v97/recht19a.html.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, April 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y. URL http://dx.doi.org/10.1007/s11263-015-0816-y.

Abdelrahman Shaker, Muhammad Maaz, Hanoona Rasheed, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. SwiftFormer: Efficient additive attention for transformer-based real-time mobile vision applications. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17379–17390, 2023. doi: 10.1109/ICCV51070.2023.01598.

Dai Shi. TransNeXt: Robust foveal visual perception for vision transformers. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17773–17783, 2024. doi: 10.1109/CVPR52733.2024.01683.

Zhenfeng Su, Kang Zhao, Han Bao, Tao Yuan, Zhongzhe Hu, Xianzhi Yu, and Wenxuan Wang. Rethinking depth pruning for vision transformers: A heterogeneity-aware perspective. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=8X5SOsxwQi.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/tan19a.html.

Yunjie Tian, Lingxi Xie, Zhaozhi Wang, Longhui Wei, Xiaopeng Zhang, Jianbin Jiao, Yaowei Wang, Qi Tian, and Qixiang Ye. Integrally pre-trained transformer pyramid networks. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18610–18620, 2023. doi: 10.1109/CVPR52729.2023.01785.

Yunjie Tian, Lingxi Xie, Jihao Qiu, Jianbin Jiao, Yaowei Wang, Qi Tian, and Qixiang Ye. Fast-iTPN: Integrally pre-trained transformer pyramid network with token migration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9766–9779, 2024. doi: 10.1109/TPAMI.2024.3429508.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24261–24272. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Paper.pdf.

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 *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 548–558, 2021. doi: 10.1109/ICCV48922.2021.00061.

Ruoyu Wu, Jingtao Liu, Yue Wang, Dongguang Li, and Derong Chen. SFViT: A compact squeeze-and-fusion vision transformer for efficient visual perception. *Journal of King Saud University Computer and Information Sciences*, 38(6), June 2026. ISSN 2213-1248. doi: 10.1007/s44443-026-00931-z. URL http://dx.doi.org/10.1007/s44443-026-00931-z.

Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. MetaFormer is actually what you need for vision. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10809–10819. IEEE, June 2022. doi: 10.1109/cvpr52688.2022.01055. URL http://dx.doi.org/10.1109/CVPR52688.2022.01055.

Xiaosong Zhang, Yunjie Tian, Lingxi Xie, Wei Huang, Qi Dai, Qixiang Ye, and Qi Tian. HiViT: A simpler and more efficient design of hierarchical vision transformer. In *International Conference on Learning Representations*, 2023.

A Additional Material

This appendix collects the detailed protocols and the secondary results that support the main paper.

A.1 Frozen-backbone probes: protocol and full results

With global average pooling P and affine classifiers A_i, the three heads are

$$H_0(Z) = A_0(P(Z)), \qquad H_1(Z) = H_{\text{native,new}}(Z), \qquad H_3(Z) = A_3(\text{LN}(P(Z))) \,. \tag{S1}$$

At each attachment point, all head parameters are fitted with cross-entropy on the frozen backbone. H_0 is affine in the pooled features, whereas H_3 is affine in their normalized counterpart, since LayerNorm depends on its input. The newly fitted H_1 keeps the native normalization and its placement.

Protocol. We probe 18 frozen backbones from five families: PoolFormer (S24, S36, M36, M48), VMamba (T, S, B), Swin (T, S, B), MogaNet (S, B, L, XL), and ConvNeXt (T, S, B, L) (Yu et al., 2022; Liu et al., 2024b; 2021; Li et al., 2024; Liu et al., 2022b). Every checkpoint was trained from scratch on ImageNet-1K without distillation. Training and validation features both use the deterministic evaluation preprocessing distributed with each model. At each attachment point, H_0 applies global average pooling (GAP) followed by a learned linear classifier; H_1 uses a freshly trained native head that keeps its normalization type and placement; H_3 applies GAP, LayerNorm ($\epsilon = 10^{-5}$), and a linear classifier (Ba et al., 2016). In some families, two of these readouts coincide. MogaNet's native head is already GAP followed by a linear classifier, so H_1 reduces to H_0 and the two differ only by head-seed noise. ConvNeXt normalizes after pooling with $\epsilon = 10^{-6}$, so its H_3 differs from its H_1 only by the epsilon and serves as a near-null control, not as an exact placement contrast. The backbone stays in evaluation mode. Heads are trained for 20 epochs with AdamW (Loshchilov & Hutter, 2019), a learning rate of 10^{-3}, zero weight decay, two warmup epochs, cosine decay to 10^{-5} (Loshchilov & Hutter, 2017), and an effective batch size of 1,024. We use three head seeds and report the mean of the best validation Top-1 scores, selected separately within each run.

Results. Table S1 reports D_3 and S_4 accuracies under the three readouts. Relative to H_0, H_3 raises the S_4 accuracy of PoolFormer-S24 from 75.82 to 79.90 and the D_3 accuracy of MogaNet-S from 70.74 to 78.30. Under H_1, the D_3-to-S_4 accuracy difference is 10.68 points for Swin-T, 2.52 for Swin-S, and 0.30 for Swin-B, and 7.13, 0.65, 0.23, and 0.17 points for ConvNeXt-T/S/B/L. Eight D_3 entries, marked † in the table, peak within the last two epochs of the head schedule. Two of the comparisons above rest on such entries: the 10.68-point Swin-T gap under H_1 and the 70.74 starting point of the MogaNet-S normalization contrast.

Removal with the pretrained head fixed. We also remove S_4 from the pretrained Swin-T and Swin-B checkpoints, keeping D_3 and all surviving weights, including the head. We evaluate on the 50,000 ImageNet validation images, resized to 256 pixels on the shorter side and center-cropped to 224×224, without updating any parameters. Top-1 falls from 81.17 to 58.43 for Swin-T and from 83.41 to 83.17 for Swin-B.

A.2 S_4 shortcut sweep

Protocol. On the 18 pretrained backbones, we vary the terminal feature map as

$$z_\alpha = u + \alpha\big(S_4(u) - u\big), \qquad u = D_3(E(x)), \tag{S2}$$

and feed z_α to the unchanged pretrained head. We evaluate $\alpha \in \{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 2\}$ on all 50,000 ImageNet-1K validation images at 224×224, with batch size 128 and no parameter updates. The blend combines the D_3 and S_4 tensors before the native head normalization: $\alpha = 0$ bypasses S_4, and $\alpha = 1$ recovers the original output. The $\alpha = 0$ column is therefore the same intervention as the fixed-weight removal of Section A.1, measured in a separate evaluation run and reported here with the opposite sign convention: Equation S4 gives full minus truncated accuracy, whereas Table S2 gives the change relative to the unmodified model.

Table S1: Best validation Top-1 (%) of frozen-backbone probes, averaged over three head seeds; Native denotes the unmodified pretrained model. H_0: GAP–linear; H_1: native head retrained from scratch; H_3: GAP–LayerNorm–linear. †: best score in the final two epochs, possibly unconverged.

	D_3			S_4			
Model	H_0	H_1	H_3	H_0	H_1	H_3	Native
PoolFormer S24	68.40†	70.45	70.43	75.82	79.98	79.90	80.33
PoolFormer S36	70.30†	72.65	72.65	77.08	81.10	81.10	81.44
PoolFormer M36	72.02	76.21	76.10	78.38	81.91	81.86	82.11
PoolFormer M48	73.07	77.24	77.12	78.90	82.37	82.32	82.45
VMamba T	74.23	74.09	73.39	81.87	82.31	82.28	82.49
VMamba S	81.78	82.06	82.01	82.75	83.57	83.56	83.64
VMamba B	82.68	83.65	83.70	82.71	83.71	83.71	83.88
Swin T	70.12	70.20†	69.86†	80.31	80.88	80.92	81.18
Swin S	80.38	80.47	80.34	82.45	82.99	82.98	83.21
Swin B	82.27	83.07	83.09	82.18	83.37	83.34	83.42
MogaNet S	70.74†	70.74†	78.30	83.17	83.19	82.88	83.36
MogaNet B	80.81†	80.81†	82.34	84.24	84.23	83.91	84.44
MogaNet L	83.53	83.53	83.08	84.52	84.54	84.18	84.71
MogaNet XL	84.29	84.32	84.13	85.10	85.10	84.51	85.14
ConvNeXt T	75.41	75.01	75.00	82.29	82.14	82.15	82.52
ConvNeXt S	82.73	82.59	82.60	83.38	83.24	83.30	83.63
ConvNeXt B	83.51	83.45	83.42	83.87	83.68	83.69	84.06
ConvNeXt L	83.90	83.53	83.54	84.02	83.70	83.75	84.41

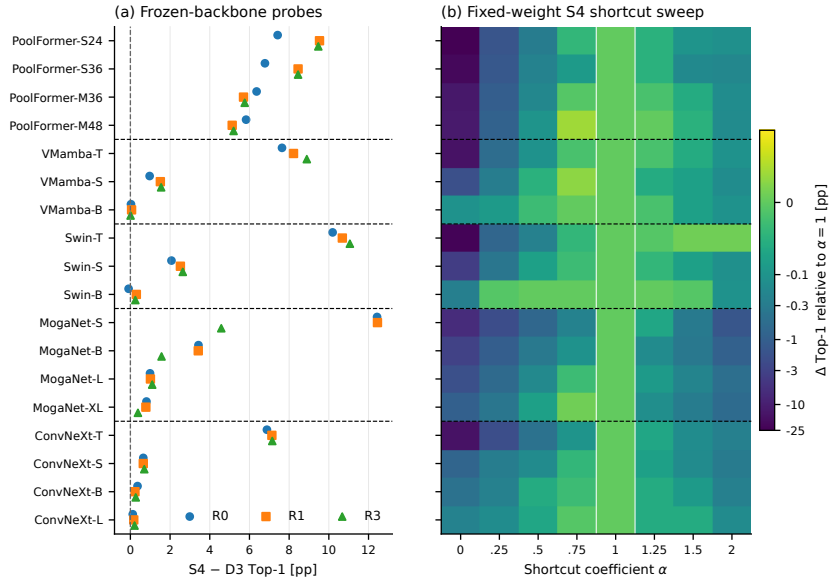


Figure S1: **Contribution of the fourth stage in pretrained backbones. (a)** Frozen-probe Top-1 gap ($S_4 - D_3$) under H_0, H_1, and H_3 (labeled R0, R1, and R3), averaged over three head seeds. **(b)** Fixed-weight shortcut sweep: Top-1 change relative to the native model ($\alpha = 1$); $\alpha = 0$ bypasses S_4. Differences are in percentage points. Complete scores are in Tables S1 and S2.

Results. Figure S1(b) and Table S2 report accuracy changes relative to the unmodified model. At $\alpha = 0$, the loss ranges from 0.11 points for VMamba-B to 24.76 points for PoolFormer-S24; at $\alpha = 0.25$, it ranges from 0.01 to 2.04 points. For Swin-T, the change is -22.77 at $\alpha = 0$, -0.74 at

Table S2: Fixed-weight S_4 shortcut sweep. Native is the unmodified model's Top-1 (%); Δ gives changes relative to Native, in percentage points.

	Δ Top-1 at shortcut coefficient α								
Model	0	0.25	0.5	0.75	1	1.25	1.5	2	Native
PoolFormer S24	−24.76	−1.50	−0.29	−0.04	+0.00	−0.05	−0.10	−0.12	80.33
PoolFormer S36	−20.03	−1.34	−0.22	−0.09	+0.00	−0.05	−0.16	−0.18	81.44
PoolFormer M36	−12.83	−1.04	−0.18	−0.01	+0.00	−0.02	−0.06	−0.14	82.11
PoolFormer M48	−11.80	−0.87	−0.10	+0.04	+0.00	−0.00	−0.05	−0.15	82.45
VMamba T	−14.68	−0.56	−0.11	−0.02	+0.00	−0.02	−0.06	−0.11	82.49
VMamba S	−1.89	−0.28	−0.05	+0.03	+0.00	−0.06	−0.08	−0.14	83.64
VMamba B	−0.11	−0.09	−0.03	−0.00	+0.00	−0.02	−0.08	−0.11	83.88
Swin T	−22.77	−0.74	−0.23	−0.04	+0.00	−0.01	+0.01	+0.01	81.18
Swin S	−3.59	−0.39	−0.08	−0.03	+0.00	−0.04	−0.08	−0.12	83.21
Swin B	−0.25	−0.01	+0.00	−0.00	+0.00	+0.00	−0.01	−0.11	83.42
MogaNet S	−6.44	−2.04	−0.71	−0.22	+0.00	−0.07	−0.33	−1.46	83.36
MogaNet B	−3.07	−1.02	−0.34	−0.11	+0.00	−0.15	−0.35	−0.91	84.44
MogaNet L	−1.90	−0.63	−0.17	−0.03	+0.00	−0.09	−0.21	−0.52	84.71
MogaNet XL	−0.96	−0.38	−0.08	+0.01	+0.00	−0.13	−0.29	−0.61	85.14
ConvNeXt T	−15.10	−1.97	−0.46	−0.10	+0.00	−0.07	−0.14	−0.24	82.52
ConvNeXt S	−0.77	−0.28	−0.15	−0.04	+0.00	−0.08	−0.15	−0.30	83.63
ConvNeXt B	−0.46	−0.23	−0.06	−0.03	+0.00	−0.06	−0.09	−0.24	84.06
ConvNeXt L	−0.22	−0.14	−0.07	−0.01	+0.00	−0.06	−0.06	−0.15	84.41

$\alpha = 0.25$, and -0.23 at $\alpha = 0.5$. At $\alpha = 2$, the largest loss is 1.46 points, for MogaNet-S, whereas the other families lose at most 0.30 points.

A.3 Training from scratch and allocation controls

Training protocol. We train full and truncated Swin and ConvNeXt models from scratch for 300 epochs with the ImageNet-1K training recipe of each family. Within a family, paired models share the data augmentation, optimizer, learning-rate schedule, and effective batch size. For a variant with N blocks, the maximum stochastic-depth probability (Huang et al., 2016) is $p(N) = p_{\text{full}}(N-1)/(N_{\text{full}}-1)$, which preserves the slope of the baseline schedule. Table 1 of the main paper reports the best validation Top-1 over 300 epochs, with non-EMA weights for ConvNeXt. Training and checkpoint-evaluation accuracies are truncated to two decimals after differences are computed.

Complete classification and resource results. Table S3 gives the absolute FLOPs, parameter counts, throughput, and memory values behind the compact table of the main paper.

Peak-memory measurements. Peak memory in Table S3 is the median peak allocated GPU memory in MiB over three FP32 inference runs at batch size 1 on an NVIDIA A100-SXM4-80GB; Δ memory is relative to the full model. We measure randomly initialized models with the same architectures, in PyTorch eager mode at 224×224. FPS is the global throughput at batch size 128 on the same GPU in FP32, also measured on randomly initialized models. The same protocol applies to all ten pairs, including Nano.

Width and stage-depth controls. Standard Tiny models use normalized depths $[1, 1, 3, 1]$, whereas Small, Base, and Tiny-C64 use $[1, 1, 9, 1]$ under the common $C, 2C, 4C, 8C$ width progression. In Swin, with about 29% of the parameters removed, the full-minus-truncated Top-1 gaps are 0.13, 0.53, and 0.79 points at base widths 128, 96, and 64. The full Tiny-C64 models use 21.3% fewer parameters than standard Tiny for Swin and 20.7% fewer for ConvNeXt, with Top-1 of 81.18 versus 81.20 for Swin and 81.69 versus 82.04 for ConvNeXt.

Table S3: Training with and without S_4 at 224×224 resolution. Best validation Top-1; Removed is the percentage of parameters removed. Δ Top-1 is truncated minus full in percentage points; Δ memory is the percentage change relative to full. GFLOPs are per image. FP32 inference uses batch size 128 for throughput and batch size one for peak allocated GPU memory.

(a) Accuracy and parameters

Model	Params (M) Full / Trunc.	Removed (%)	Top-1 (%) Full / Trunc.	Δ Top-1 (pp)
Swin-B	87.7 / 62.5	28.7%	83.33 / 83.19	−0.13
Swin-S	49.6 / 35.4	28.6%	83.04 / 82.50	−0.53
Swin-T	28.2 / 14.1	50.1%	81.20 / 78.78	−2.42
Swin-T-C64	22.2 / 15.9	28.4%	81.18 / 80.39	−0.79
Swin-Nano-C32	5.7 / 4.1	27.6%	73.84 / 70.98	−2.86
ConvNeXt-B	88.5 / 63.2	28.6%	83.84 / 83.39	−0.44
ConvNeXt-S	50.2 / 35.9	28.5%	83.23 / 82.91	−0.31
ConvNeXt-T	28.5 / 14.3	50.0%	82.04 / 80.65	−1.38
ConvNeXt-T-C64	22.6 / 16.2	28.2%	81.69 / 81.32	−0.36
ConvNeXt-Nano-C32	4.1 / 3.0	26.3%	74.32 / 71.24	−3.08

(b) Computation and memory

Model	FLOPs (G) Full / Trunc.	Throughput (img/s) Full / Trunc.	Peak mem. (MiB) Full / Trunc.	Δ mem. (%)
Swin-B	15.43 / 14.19	447.4 / 476.4	374.7 / 277.6	−25.9%
Swin-S	8.74 / 8.04	704.2 / 753.2	218.9 / 164.7	−24.8%
Swin-T	4.49 / 3.79	1232.2 / 1379.1	137.1 / 81.8	−40.4%
Swin-T-C64	3.94 / 3.62	1192.8 / 1256.9	107.2 / 82.5	−23.0%
Swin-Nano-C32	1.03 / 0.95	2412.0 / 2502.4	37.9 / 31.8	−16.0%
ConvNeXt-B	15.37 / 14.13	454.2 / 489.8	362.0 / 265.3	−26.7%
ConvNeXt-S	8.70 / 8.00	719.3 / 769.9	214.4 / 159.8	−25.4%
ConvNeXt-T	4.46 / 3.76	1268.6 / 1430.3	131.8 / 75.6	−42.6%
ConvNeXt-T-C64	3.91 / 3.60	1223.2 / 1293.9	103.7 / 78.5	−24.3%
ConvNeXt-Nano-C32	0.68 / 0.63	4059.7 / 4243.0	28.2 / 24.1	−14.6%

Reallocation at the standard Tiny budget. For the standard Tiny models ($C = 96$), reallocating the terminal budget to S_3 gives depths $[2, 2, 14]$ for Swin and $[3, 3, 21]$ for ConvNeXt, while retaining D_3 and the native head. These variants have 28,316,434 and 28,722,760 parameters and require 6.628 and 6.590 GFLOPs per 224×224 image, respectively.

Reduced-width Swin controls. We train five $C = 64$ variants, including the full baseline, for 300 epochs with the stochastic-depth slope $0.2/23$ of the full $[2, 2, 18, 2]$ model. Table S4 reports their best validation epoch. Removing S_4 costs 0.79 points. At nearly the same parameter count, the three-stage $[2, 2, 18]$ allocation exceeds the four-stage $[2, 2, 10, 2]$ control by 0.41 points. Replacing the last two local S_3 blocks with global-attention blocks adds 0.16 points, and deepening S_3 to 23 blocks reduces the gap to the full model to 0.46 points, with 12.2% more FLOPs than the full model.

Allocation controls in both families. Table S5 compares three-stage and four-stage allocations at approximately matched parameter counts for the Tiny-C64 models. For Swin, the three-stage $[2, 2, 18]$ model reaches 80.39 Top-1 at 15.948M parameters, versus 79.98 at 15.929M for the four-stage $[2, 2, 10, 2]$ control. For ConvNeXt, the corresponding $[3, 3, 27]$ and $[3, 3, 15, 3]$ allocations reach 81.32 and 81.02 at 16.289M and 16.200M parameters, respectively. With the stochastic-depth slope shared within each family, the three-stage allocations exceed the four-stage controls by 0.41 and 0.30 points; these comparisons are not FLOP-matched.

Table S4: Reduced-width Swin controls ($C = 64$). Global S_3 replaces the final two local S_3 attention blocks.

Variant	Built depths	Params (M)	GFLOPs	Top-1	Top-5
Full	$[2, 2, 18, 2]$	22.258	3.943	81.18	95.67
Truncated, local	$[2, 2, 18]$	15.948	3.629	80.39	95.21
Four-stage control	$[2, 2, 10, 2]$	15.929	2.670	79.98	95.13
Truncated, deeper	$[2, 2, 23]$	19.903	4.425	80.72	95.37
Truncated, global S_3	$[2, 2, 18]$	15.957	3.659	80.56	95.30

Table S5: Tiny-C64 allocation controls. Four-stage controls approximately match the truncated parameter budget. Changes are relative to each model's full baseline in Table 1 of the main paper (Top-1 in percentage points). FPS follows the measurement protocol of Table S3.

Model	Variant	Params (M)	Δ params	Top-1	Δ Top-1	FPS
Swin-T-C64	Truncated	15.948	-28.4%	80.39	-0.79	1256.9
	Four-stage S_4 control	15.929	-28.4%	79.98	-1.20	1607.5
ConvNeXt-T-C64	Truncated	16.289	-28.2%	81.32	-0.36	1293.9
	Four-stage S_4 control	16.200	-28.5%	81.02	-0.67	1663.2

Nano controls ($C = 32$). The Nano models of both families use widths $[32, 64, 128, 256]$, retain D_3 and the 256-dimensional classification head, and disable stochastic depth in all variants. Each run completes 300 epochs with its family recipe. Table S6 reports the best validation Top-1, with non-EMA weights for ConvNeXt. The four-stage controls exceed the truncated models by 0.85 points for Swin and 1.08 for ConvNeXt with slightly fewer parameters. Reallocating the full budget to S_3 still leaves gaps of 0.32 and 0.78 points, despite slightly more parameters and higher GFLOPs.

A.4 Paired prediction analysis

Nine of the ten pairs have a nominal Top-1 $p < 0.05$, whereas Swin-B has $p = 0.228$. The tests use the selected checkpoints, and the p-values are not adjusted for checkpoint selection or multiple comparisons.

For all ten full/truncated pairs (Tiny, Tiny-C64, Small, Base, and Nano-C32), we evaluate both selected checkpoints on the same 50,000 ImageNet-1K validation images, with the selected Swin weights and the non-EMA ConvNeXt weights. Inputs are resized to 256 pixels, center-cropped to 224×224, and evaluated with CUDA mixed precision. For Top-1 and Top-5 separately, let n_F count the images that only the full model classifies correctly and n_T those that only the truncated model classifies correctly. The accuracy difference, in percentage points, is $100(n_F - n_T)/50{,}000$. The two-sided exact McNemar test computes the p-value with a binomial test on the $n_F + n_T$ discordant images, under equal probabilities of full-only and truncated-only correctness.

These scores come from fresh evaluations of the saved checkpoints, whereas Table 1 of the main paper reports training-log maxima. For ConvNeXt-T, the logs give 82.04/80.65 (full/truncated), versus 82.04/80.59 here; the cause of this difference remains undetermined.

The full checkpoint has more correct-only images in every pair, although for Swin-B the net excess is only 64 of 2,728 discordant Top-1 images. Top-5 p-values exceed 0.05 for Swin-S, Swin-B, and ConvNeXt-B. These tests characterize the selected checkpoint pairs; they do not estimate training variability.

A.5 Representation-analysis protocol and terminal ERFs

Representation-analysis protocol. We analyze the ten full/truncated checkpoint pairs, including the Tiny-C64 ($C = 64$) and Nano-C32 ($C = 32$) controls, with the selected Swin weights and the

Table S6: Swin-Nano and ConvNeXt-Nano training and allocation controls at $C = 32$. Best non-EMA validation Top-1 over 300 epochs at 224×224; accuracies are truncated to two decimals.

Family	Variant	Depths	Params (M)	GFLOPs	Top-1 (%)
Swin	Full	$[2, 2, 18, 2]$	5.723	1.027	73.84
	Truncated	$[2, 2, 18]$	4.140	0.948	70.98
	Reallocated	$[2, 2, 26]$	5.732	1.276	73.52
	Four-stage control	$[2, 2, 10, 2]$	4.131	0.699	71.83
ConvNeXt	Full	$[2, 2, 18, 2]$	4.096	0.683	74.32
	Truncated	$[2, 2, 18]$	3.018	0.630	71.24
	Reallocated	$[2, 2, 26]$	4.126	0.847	73.54
	Four-stage control	$[2, 2, 10, 2]$	2.988	0.467	72.33

Table S7: Paired ImageNet-1K predictions: full-minus-truncated accuracy differences (pp) and two-sided exact McNemar tests. n_F/n_T count Top-1 images correct only for the full/truncated model. p-values are unadjusted for checkpoint selection or multiple comparisons.

	Top-1				Top-5	
Model	Δ (pp)	n_F	n_T	Exact p	Δ (pp)	Exact p
Swin-T	2.41	2649	1442	$1.85 \cdot 10^{-80}$	1.00	$5.11 \cdot 10^{-37}$
Swin-T-C64	0.79	1956	1559	$2.29 \cdot 10^{-11}$	0.46	$2.69 \cdot 10^{-11}$
Swin-S	0.53	1684	1416	$1.60 \cdot 10^{-6}$	0.12	0.0554
Swin-B	0.12	1396	1332	0.228	0.06	0.330
Swin-Nano-C32	2.83	3680	2262	$3.53 \cdot 10^{-76}$	1.48	$1.29 \cdot 10^{-49}$
ConvNeXt-T	1.44	2471	1749	$9.38 \cdot 10^{-29}$	0.45	$3.91 \cdot 10^{-9}$
ConvNeXt-T-C64	0.36	2018	1834	0.00319	0.39	$1.10 \cdot 10^{-7}$
ConvNeXt-S	0.31	1598	1440	0.00439	0.16	0.0107
ConvNeXt-B	0.44	1532	1312	$3.98 \cdot 10^{-5}$	0.05	0.382
ConvNeXt-Nano-C32	3.08	3686	2142	$6.61 \cdot 10^{-92}$	1.44	$3.77 \cdot 10^{-47}$

non-EMA ConvNeXt weights. Inputs are resized to 256 pixels on the shorter side, center-cropped to 224×224, and normalized with ImageNet statistics. Linear centered kernel alignment (CKA) (Kornblith et al., 2019) is computed on GAP features from all 50,000 validation images, centered across images. For all ten pairs, we also measure the effective receptive field (ERF) (Luo et al., 2016) using eight Rademacher channel projections per image and attachment point, at a single probe unit near the center of each grid. Input-gradient maps are squared and averaged over input channels, projections, and images before the square root is taken. We report r_{50}, the radius in input pixels that contains half of the map mass within the largest disk centered on the probe and contained in the image.

CKA at all shared attachment points. Table S8 reports all stagewise measurements. Each comparison uses the same ordered validation images and matching attachment points within one model family and scale. Similarity is high at D_2 for all ten pairs but not uniformly high at earlier points, particularly at S_1 for ConvNeXt, so the stagewise values do not form a monotone trajectory.

Terminal effective receptive fields. Comparing truncated D_3 with full S_4 at the terminal output, r_{50} is 62.837 against 60.121 pixels for Swin-S and 59.114 against 54.557 for Swin-B. The corresponding values are 57.866 against 55.322 for ConvNeXt-S and 57.136 against 53.390 for ConvNeXt-B. For Swin-T and Swin-T-C64, truncated D_3 measures 59.266 against 72.584 and 57.276 against 69.588 pixels for full S_4. For ConvNeXt-T and ConvNeXt-T-C64, the comparisons are 57.658 against 57.658 and 58.588 against 62.933 pixels. For Swin-Nano and ConvNeXt-Nano, truncated D_3 measures 52.844 against 68.909 and 52.540 against 54.722 pixels for full S_4.

Table S8: Linear CKA between independently trained full and truncated checkpoints at shared attachment points, using pooled features from all 50,000 ImageNet validation images.

Model	Stem	S_1	D_1	S_2	D_2	S_3	D_3
Swin-T	0.999	0.996	0.993	0.974	0.977	0.436	0.362
Swin-T-C64	0.998	0.997	0.988	0.979	0.996	0.241	0.188
Swin-S	0.996	0.998	0.954	0.898	0.996	0.373	0.467
Swin-B	0.999	0.998	0.994	0.981	0.997	0.606	0.520
Swin-Nano-C32	0.992	0.990	0.995	0.991	0.993	0.163	0.236
ConvNeXt-T	0.966	0.751	0.968	0.872	0.975	0.432	0.078
ConvNeXt-T-C64	0.966	0.646	0.919	0.913	0.979	0.545	0.173
ConvNeXt-S	0.996	0.878	0.971	0.898	0.983	0.675	0.275
ConvNeXt-B	0.972	0.879	0.844	0.946	0.989	0.736	0.301
ConvNeXt-Nano-C32	0.931	0.848	0.968	0.723	0.941	0.485	0.106

Spatial-frequency spectrum (exploratory). We also compare the spatially mean-centered FFT power of full and truncated features at matching attachment points. At D_3, the fraction in the upper frequency band (normalized radial frequency 0.5–1) decreases for all five non-EMA ConvNeXt pairs (by 0.088–0.234), whereas it increases for Swin-T-C64/S/B (by 0.042–0.278) and decreases for Swin-T and Swin-Nano (by 0.198 and 0.143).

A.6 Block-wise readout profiles

Protocol. We analyze the ten full/truncated pairs, the parameter-matched reallocated Swin-T, and the Nano-C32 reallocated ($[2, 2, 26]$) and four-stage ($[2, 2, 10, 2]$) controls of both families (selected Swin weights, non-EMA ConvNeXt weights). Features are globally average-pooled after D_2, after every S_3 block, at D_3, and after every S_4 block. H_3 heads are fitted at every point and H_0 heads at the stage boundaries, on all 1,281,167 ImageNet-1K training images without augmentation, using the recipe of Section A.1 (20 epochs, AdamW, learning rate 10^{-3}, zero weight decay, two warmup epochs, cosine decay, batch 1,024). We report final-epoch accuracy on the 50,000 validation images, without best-epoch selection. Δ_{prog} follows Eq. equation 3 of the main paper. Image-level 95% intervals use a normal approximation stratified by class and are conditional on the checkpoints and heads; they do not measure variability between training runs.

Results. Figure S2 shows all ten profiles. Table S9 reports Δ_{prog} with the differences at D_2, at the end of S_3, and at the block where the gap peaks. Table S10 gives the boundary readouts under both heads, and Table S11 the H_3 differences between attachment points. Under H_0, the truncated models also exceed their full counterparts at the end of S_3 in nine pairs, the exception being Swin-B, although H_0 scores are sensitive to head parameterization (Section A.7).

For the Nano-C32 allocation controls, end-of-S_3 H_3 accuracies are 71.56 and 73.51 for the truncated $[2, 2, 18]$ and reallocated $[2, 2, 26]$ Swin models, versus 57.02 and 51.59 for the full $[2, 2, 18, 2]$ and four-stage $[2, 2, 10, 2]$ models. The corresponding ConvNeXt scores are 70.10 and 72.40 versus 55.25 and 51.83. In each family, both models without S_4 exceed both models with S_4 by at least 14.5 points.

Boundary readouts and the contribution measured across S_4. Under H_3, the gain from the end of S_3 to the final S_4 block of full models falls from 16.76 (Swin) and 18.75 (ConvNeXt) points at Nano-C32 to -0.18 and -0.54 at Base, and it is smaller for T-C64, whose S_3 is deeper, than for T (Figure 2h of the main paper). Measured from D_3, S_4 still adds 0.34 and 0.90 points at Base, which partly offsets the drop that D_3 introduces under H_3. From Small upward, truncated D_3 readouts match or exceed full S_4 readouts, by 0.18–0.38 points, whereas at Nano-C32 they remain 2.09–2.36 points below. The effect of D_3 itself depends on the head and on scale: under H_3, it adds 2.28–7.07 points in the full ConvNeXt Nano-C32, T, and T-C64 models but lowers accuracy by 0.52–1.44 points in the full Small and Base models of both families. Normalizing before pooling

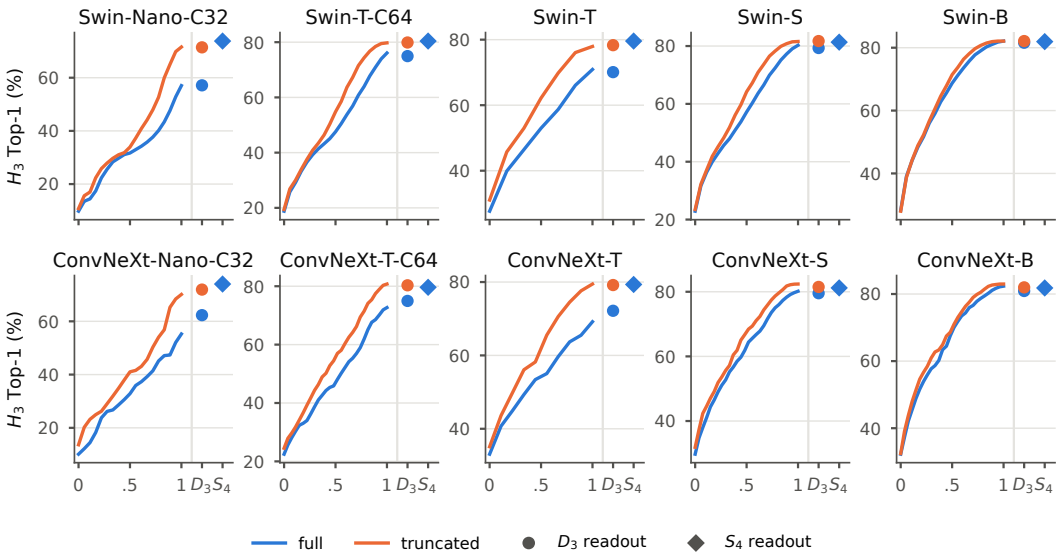


Figure S2: H_3 Top-1 along relative depth in S_3 for all ten full/truncated pairs; markers give the D_3 and final-S_4 readouts.

matters mainly in the small full models, in opposite directions for the two families: each S_3 is better decoded with the normalization scope of its own D_3 (Section A.7).

Similarity and pooled-feature geometry. Block-by-block linear CKA on the validation features refines the stagewise comparison. For Swin Nano-C32 to Small, truncated D_3 is more similar to the last full S_4 block (0.588–0.755) than to full D_3 (0.188–0.467); for Swin-B both values are 0.520. For ConvNeXt, the corresponding values are lower (0.147–0.334 against 0.078–0.301), and the final S_4 block is the closer of the two in four of five pairs. The raw ConvNeXt features at the end of S_3 have a low effective rank (41–145, against 119–449 after D_3), so raw rank and CKA are not compared across families. Among the geometric statistics, the between-class variance fraction at the end of S_3 is higher in the truncated model for all Swin pairs and for ConvNeXt Nano-C32, T-C64, and T, but lower for ConvNeXt-S and ConvNeXt-B. The final S_4 block lowers the relative spatial variance η of full models by 0.15–0.56 in Swin and by 0.07–0.12 in ConvNeXt T-C64 to B, whereas it slightly raises η for ConvNeXt-Nano ($+0.03$). Truncated Swin D_3 features reach η values of 0.16–0.30 from Nano-C32 to Small, close to those after full S_4, and their effective rank stays below that of full S_4 in all ten pairs.

A.7 Operations inside D_3

Protocol. On the same frozen checkpoints, we decompose D_3 with pooled views of the last S_3 output z. Each 2×2 patch of z is concatenated into a $4C$ vector, and these vectors are averaged over patches (raw patch view). Two normalizations without affine parameters and with a common $\epsilon = 10^{-6}$ are applied before pooling: per token over C channels, or per concatenated patch over $4C$ channels. Two controls have the same dimension: a fixed, label-independent partition of the 14×14 positions into four balanced groups, and the global average of z repeated four times. The primary contrast, fixed before the analysis, is $E_{\text{norm}} = \text{Acc}_{H_0}(\text{token}) - \text{Acc}_{H_0}(\text{patch})$, with H_3 as a sensitivity check. Because the native projection is affine and shared across patches, pooling commutes with it. No part of D_3 is retrained.

Results. Table S12 reports E_{norm}. The token-normalized view is less decodable than the patch-normalized view for all ten Swin models and more decodable for nine of the ten ConvNeXt models, with the largest effects in the full Nano-C32, T, and T-C64 ConvNeXt models. The effect shrinks in the truncated models and at larger scale, and several small values change sign under H_3. Each family's full S_3 is better decoded with the normalization scope of its own D_3 (the concatenated patch for Swin, the token for ConvNeXt). The 2×2 patch structure improves H_0 accuracy over the

Table S9: Differences in H_3 Top-1 (percentage points) between a comparator and a reference checkpoint along S_3. The 95% intervals for Δ_{prog} are computed over images, conditional on the checkpoints and fitted heads. Reallocated denotes the parameter-matched truncated model with a deeper S_3; four-stage denotes the Nano-C32 $[2, 2, 10, 2]$ control.

Model	Contrast	Δ_{prog}	95% interval	$\delta(D_2)$	δ end of S_3	Peak δ (block)
Swin-Nano-C32	truncated − full	+5.79	[5.57, 6.02]	+0.98	+14.53	+17.39 (16/18)
Swin-T-C64	truncated − full	+4.75	[4.51, 4.98]	+0.62	+3.66	+10.73 (13/18)
Swin-T	truncated − full	+4.58	[4.27, 4.89]	+3.41	+7.02	+11.13 (4/6)
Swin-S	truncated − full	+3.61	[3.36, 3.86]	+0.59	+1.31	+7.27 (11/18)
Swin-B	truncated − full	+1.55	[1.29, 1.81]	−0.24	−0.01	+2.76 (11/18)
ConvNeXt-Nano-C32	truncated − full	+4.51	[4.24, 4.78]	+3.53	+14.85	+17.91 (16/18)
ConvNeXt-T-C64	truncated − full	+4.64	[4.35, 4.93]	+2.03	+8.07	+10.48 (20/27)
ConvNeXt-T	truncated − full	+5.74	[5.42, 6.06]	+2.02	+10.27	+12.11 (8/9)
ConvNeXt-S	truncated − full	+1.39	[1.09, 1.70]	+2.12	+2.23	+5.30 (12/27)
ConvNeXt-B	truncated − full	+1.81	[1.51, 2.11]	+0.18	+0.63	+4.07 (9/27)
Swin-Nano-C32	reallocated − full	+7.09	[6.88, 7.31]	+0.34	+16.49	—
ConvNeXt-Nano-C32	reallocated − full	+6.25	[6.00, 6.50]	+2.08	+17.15	—
Swin-Nano-C32	four-stage − full	−3.18	[−3.40, −2.97]	+1.00	−5.44	—
ConvNeXt-Nano-C32	four-stage − full	−6.88	[−7.17, −6.60]	+6.85	−3.42	—
Swin-Nano-C32	truncated − four-stage	+8.98	[8.75, 9.21]	−0.02	+19.97	—
ConvNeXt-Nano-C32	truncated − four-stage	+11.39	[11.12, 11.66]	−3.32	+18.27	—
Swin-T	reallocated − full	+12.19	[11.91, 12.48]	−1.80	+10.11	—

Table S10: Final-epoch validation Top-1 (%) of heads fitted on frozen checkpoints at the end of S_3, at D_3, and after the last S_4 block.

	H_3					H_0			
	Full			Truncated		Full		Truncated	
Model	S_3 end	D_3	S_4	S_3 end	D_3	S_3 end	D_3	S_3 end	D_3
Swin-Nano-C32	57.02	57.14	73.79	71.56	71.43	55.20	52.12	71.42	71.26
Swin-T-C64	76.09	74.96	80.40	79.75	79.89	76.12	75.59	79.52	79.33
Swin-T	70.91	70.09	79.57	77.93	78.32	70.87	70.37	77.88	77.20
Swin-S	80.27	79.32	81.34	81.57	81.72	80.02	79.96	80.77	80.53
Swin-B	82.11	81.59	81.94	82.10	82.11	81.16	80.68	80.83	80.29
ConvNeXt-Nano-C32	55.25	62.32	74.00	70.10	71.92	57.46	60.78	70.45	71.77
ConvNeXt-T-C64	72.69	74.97	79.63	80.75	80.32	73.06	75.66	80.02	80.54
ConvNeXt-T	69.18	72.17	79.34	79.46	79.18	70.02	72.89	78.48	79.09
ConvNeXt-S	80.16	79.55	81.20	82.39	81.49	78.79	80.44	81.03	81.79
ConvNeXt-B	82.34	80.91	81.80	82.98	81.98	80.24	81.39	81.13	82.13

random partition by at most 0.22 points (range -0.52 to $+0.22$ over the 15 checkpoints with local heads). The repeated global average spans exactly the same linear functions as the global average,

Table S11: Differences in H_3 Top-1 (percentage points) between attachment points of the full (F) and truncated (T) models.

Model	$S_4 - S_3^{\text{end}}$ (F)	$S_4 - D_3$ (F)	D_3(T)$-S_4$(F)	$D_3 - S_3^{\text{end}}$ (F)	$D_3 - S_3^{\text{end}}$ (T)
Swin-Nano-C32	$+16.76$	$+16.65$	-2.36	$+0.12$	-0.13
Swin-T-C64	$+4.32$	$+5.45$	-0.52	-1.13	$+0.14$
Swin-T	$+8.66$	$+9.48$	-1.25	-0.82	$+0.39$
Swin-S	$+1.08$	$+2.02$	$+0.38$	-0.95	$+0.15$
Swin-B	-0.18	$+0.34$	$+0.18$	-0.52	$+0.01$
ConvNeXt-Nano-C32	$+18.75$	$+11.69$	-2.09	$+7.07$	$+1.82$
ConvNeXt-T-C64	$+6.95$	$+4.67$	$+0.68$	$+2.28$	-0.44
ConvNeXt-T	$+10.16$	$+7.17$	-0.16	$+2.98$	-0.28
ConvNeXt-S	$+1.04$	$+1.64$	$+0.30$	-0.61	-0.90
ConvNeXt-B	-0.54	$+0.90$	$+0.18$	-1.44	-0.99

Table S12: E_{norm}, token minus patch normalization before pooling, in percentage points: H_0 (primary) and, in parentheses, H_3.

	Swin		ConvNeXt	
Size	Full	Truncated	Full	Truncated
Nano-C32	-0.31 (-0.30)	-0.04 (-0.04)	$+1.92$ ($+0.97$)	$+0.20$ ($+0.09$)
T-C64	-0.30 (-0.34)	-0.05 ($+0.07$)	$+0.67$ ($+0.43$)	$+0.05$ (-0.02)
T	-0.31 (-0.24)	-0.06 (-0.09)	$+1.21$ ($+0.56$)	$+0.04$ (-0.01)
S	-0.13 (-0.11)	-0.14 (-0.04)	$+0.19$ (-0.00)	$+0.04$ ($+0.03$)
B	-0.07 ($+0.03$)	-0.16 (-0.02)	$+0.04$ ($+0.02$)	-0.09 (-0.11)

yet its H_0 accuracy differs by -1.86 to $+2.74$ points. H_3 reduces this discrepancy to at most 0.85 points.

A.8 Fixed-weight spatial restrictions

In Swin, we remove the window shift in the last $k \in \{2, 4, 8\}$ blocks of S_3 and, in full models, restrict S_4 attention to its diagonal. In ConvNeXt, we mask the 7×7 depthwise kernels to the central 3×3 taps in the last k blocks of S_3 and to the central tap in S_4. All trained weights are kept.

Table S13 lists every intervention on the 20 full/truncated checkpoints, evaluated on the 50,000 validation images. We use values of k up to the depth of S_3. The truncated model is more sensitive than its full counterpart in 27 of 29 S_3 conditions; the exceptions are ConvNeXt-Nano with $k = 4$ and $k = 8$. For Swin-B, ConvNeXt-S, and ConvNeXt-B, restricting S_4 costs more than bypassing it.

Terminal restrictions and token order. Removing token mixing from S_4 lowers full-model Top-1 by 39.17 points for Swin-Nano-C32 and 0.35 for Swin-B, and by 16.81 and 0.89 for the ConvNeXt counterparts. In Swin-T, the restriction costs 19.45 points versus 22.53 for bypassing S_4. Randomly permuting the 7×7 tokens before S_4 costs at most 0.55 points in Swin-T, T-C64, S, and B. Figure S3a,b plots the spatial restrictions, and panel c shows the factorial described below.

A.9 Short factorial training

The 2×2 design crosses window shifts in S_3 (on/off) with two terminal stages of nearly equal parameter count: S uses the two native S_4 blocks, while C removes token mixing. At epoch 10,

Table S13: Top-1 drop (percentage points, native minus intervention) after fixed-weight restrictions, with a separate intervention for each column. Swin removes window shifts in S_3 and restricts S_4 attention to the diagonal; ConvNeXt masks depthwise kernels to 3×3 in S_3 and to the central tap in S_4.

	Full, last k S_3 blocks			Truncated, last k S_3 blocks			Full
Model	$k = 2$	$k = 4$	$k = 8$	$k = 2$	$k = 4$	$k = 8$	S_4
Swin-Nano-C32	0.10	0.19	0.60	0.66	1.57	3.58	39.17
Swin-T-C64	0.05	0.01	0.56	0.23	0.77	1.94	7.90
Swin-T	0.11	0.52	—	0.55	1.77	—	19.45
Swin-S	−0.01	0.14	0.51	0.15	0.53	1.43	3.06
Swin-B	−0.01	0.03	0.41	0.08	0.30	1.19	0.35
ConvNeXt-Nano-C32	1.11	5.74	16.10	5.57	4.62	15.03	16.81
ConvNeXt-T-C64	1.31	1.48	2.82	5.58	4.05	5.17	8.89
ConvNeXt-T	2.60	3.09	4.83	3.96	5.08	7.30	9.65
ConvNeXt-S	0.20	0.25	0.79	0.64	2.93	3.07	3.22
ConvNeXt-B	0.11	0.30	0.73	0.12	0.79	1.88	0.89

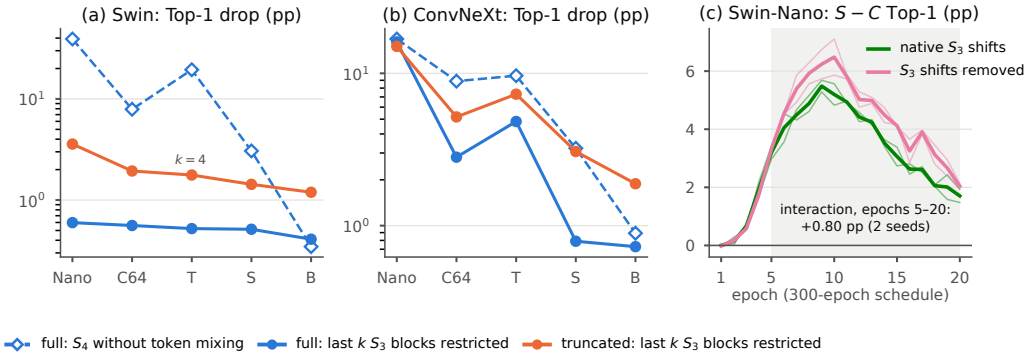


Figure S3: **Terminal token mixing and S_3 exchanges.** (a,b) Top-1 drop (log scale) after fixed-weight restrictions: S_4 without token mixing in full models, and the last k S_3 blocks restricted in full and truncated models ($k = 8$; $k = 4$ for Swin-T). Swin removes window shifts in S_3 and restricts S_4 attention to the diagonal; ConvNeXt masks depthwise kernels to 3×3 in S_3 and to the central tap in S_4. C64 denotes T-C64. (c) Swin-Nano-C32 short factorial: Top-1 advantage of the spatial terminal stage S over the token-wise stage C (percentage points). Green denotes native S_3 window shifts; pink denotes removed shifts. Thin lines show individual seeds and thick lines their two-seed mean. Shading marks epochs 5–20, over which the pink-minus-green difference defines the mean interaction.

S exceeds C by 4.83–7.11 Top-1 points across shift settings and seeds. The mean interaction over epochs 5–20 is $+0.80$ points ($+0.65$ and $+0.96$ for the two seeds).

The factorial uses Swin-Nano-C32 with depths $[2, 2, 18, 2]$, width $C = 32$, and no stochastic depth. The no-shift condition disables all S_3 window shifts; the terminal blocks are defined in Section A.17. Each run uses a global batch of 1,024 images and follows the native 300-epoch learning-rate schedule with its 20-epoch warmup, and stops after 20 epochs. For each backbone seed, the four conditions share the initialization of the common layers, the data order, the augmentations, and Mixup/CutMix draws. Table S14 reports validation Top-1 at epochs 10 and 20 and the prespecified interaction. All eight runs complete the 20 epochs; the interaction is positive for both seeds.

Table S14: Swin-Nano-C32 short factorial: validation Top-1 (%) and the interaction $[\mathrm{Acc}(S) - \mathrm{Acc}(C)]_{\text{no shift}} - [\mathrm{Acc}(S) - \mathrm{Acc}(C)]_{\text{shift}}$ averaged over epochs 5–20 (percentage points). S is the native spatial terminal stage; C is the token-wise control.

		Seed 0		Seed 1	
S_3 shifts	Terminal	Epoch 10	Epoch 20	Epoch 10	Epoch 20
native	S	30.10	46.96	30.52	46.77
native	C	25.28	45.48	24.96	44.84
removed	S	29.62	46.49	29.88	46.48
removed	C	23.76	44.53	22.77	44.35
Interaction, epochs 5–20		$+0.65$		$+0.96$	

Table S15: Differences in validation Top-1 (percentage points) between training runs at equal epochs. Final denotes epoch 300.

Model	Contrast	Epoch 5	Epoch 10	Epoch 20	Final
Swin-Nano-C32	full − truncated	$+4.29$	$+6.08$	$+4.66$	$+2.78$
Swin-T-C64	full − truncated	$+3.54$	$+4.26$	$+3.53$	$+0.76$
Swin-T	full − truncated	$+4.76$	$+5.92$	$+6.11$	$+2.49$
Swin-S	full − truncated	$+3.35$	$+2.62$	$+1.65$	$+0.65$
Swin-B	full − truncated	-0.76	-1.32	$+0.87$	$+0.23$
ConvNeXt-Nano-C32	full − truncated	$+3.57$	$+5.51$	$+4.54$	$+3.07$
ConvNeXt-T-C64	full − truncated	$+1.07$	$+4.65$	$+2.18$	$+0.40$
ConvNeXt-T	full − truncated	$+3.49$	$+6.55$	$+4.27$	$+1.40$
ConvNeXt-S	full − truncated	$+1.12$	$+2.79$	$+1.67$	$+0.30$
ConvNeXt-B	full − truncated	$+2.12$	$+2.37$	$+1.51$	$+0.43$
Swin-Nano-C32	full − reallocated	$+4.15$	$+6.68$	$+4.33$	$+0.25$
Swin-Nano-C32	four-stage − truncated	$+3.90$	$+5.70$	$+3.84$	$+0.82$
Swin-T	full − reallocated	$+4.65$	$+4.63$	$+2.21$	-0.67
Swin-T-C64	four-stage − truncated	$+4.26$	$+5.40$	$+2.15$	-0.40
Swin-T-C64	global S_3 − truncated	$+1.20$	$+1.79$	$+0.41$	$+0.21$
ConvNeXt-Nano-C32	full − reallocated	$+1.41$	$+3.30$	$+2.68$	$+0.74$
ConvNeXt-Nano-C32	four-stage − truncated	$+0.80$	$+0.93$	$+1.50$	$+1.07$
ConvNeXt-T	full − reallocated	-1.26	$+0.50$	$+0.91$	-0.20
ConvNeXt-T-C64	four-stage − truncated	$+0.09$	$+3.18$	$+1.19$	-0.27

A.10 Training curves

At epoch 10, the full-minus-truncated gap exceeds the final-epoch gap in nine of the ten pairs; Swin-B, where the truncated model leads early, is the exception.

Table S15 compares validation Top-1 from the training logs at fixed epochs. Final values are those of the last epoch, not the best checkpoints of Table 1 of the main paper. Only runs that completed 300 epochs are included. These comparisons are made at equal numbers of epochs, although the variants differ in parameters, depth, FLOPs, or stochastic depth.

A.11 Pooling order on frozen features

Frozen-backbone protocol. At D_3, we compare a post-pooling MLP, $\mathrm{GAP} \to \mathrm{Linear}(C, 4C) \to \mathrm{GELU} \to \mathrm{Linear}(4C, 1000)$, with a pre-pooling MLP that moves GAP immediately after GELU (Hendrycks & Gimpel, 2016). Both heads have identical parameter counts and paired initializations. Swin keeps its native per-position LayerNorm in both conditions (reference R2), while ConvNeXt uses no normalization in either condition (reference R4). The pre-pooling operator acts indepen-

Table S16: Pre-pooling minus post-pooling Top-1 at D_3 (percentage points): mean $\pm$ half-range over three paired head seeds. Best denotes peak validation accuracy; final denotes epoch 20.

Model	Post-pooling reference	Δ best	Δ final
Swin-T	R2, native normalization	$+3.83_{\pm 0.03}$	$+4.14_{\pm 0.23}$
Swin-S	R2, native normalization	$+0.49_{\pm 0.03}$	$+0.15_{\pm 0.08}$
ConvNeXt-T	R4, no normalization	$+1.39_{\pm 0.05}$	$+1.64_{\pm 0.03}$
ConvNeXt-S	R4, no normalization	$-0.09_{\pm 0.05}$	$-0.17_{\pm 0.07}$

dently at each spatial position. We reuse the frozen checkpoints and the head-training recipe given earlier in this appendix, with three paired head seeds. The pre-pooling code stores spatial features in float32; the reference extractor stores pooled features (and Swin normalization statistics) in float16. Both head trainers read their inputs in float32, which does not undo the reference-cache quantization. The comparison therefore changes cache precision as well as pooling order. We report both best-validation and final-epoch results.

Frozen-backbone results. Moving the MLP before pooling improves both best-validation and final-epoch Top-1 on the Tiny backbones, with smaller changes at Small scale (Table S16).

A.12 Lightweight-stage training protocol

On Swin-T-C64, with full-model depths $[2, 2, 18, 2]$, we replace S_4 and train the whole network from scratch for 300 epochs with the Swin recipe. We compare a global token-MLP plus channel-MLP (Tolstikhin et al., 2021) against a channel-only control of approximately the same parameter count, both defined in Section A.17. The two-block replacements use the full model's stochastic-depth schedule $p_i = 0.2i/23$. The 22-block truncated reference preserves this slope, with a maximum rate of $0.2 \times 21/23$.

A.13 ImageNet-C: protocol and full results

Protocol. We evaluate all ten full/truncated pairs on ImageNet-C (Hendrycks & Dietterich, 2019): Swin and ConvNeXt T/T-C64/S/B/Nano-C32. The benchmark has 15 corruption types, five severity levels, and 50,000 images per corruption and severity split, for a total of 3.75 million images per model. The additional development corruptions are excluded. Corrupted images are already 224×224 and receive only tensor conversion and ImageNet normalization. Each batch is evaluated by both members of a pair in FP32. We use the selected Swin weights and the non-EMA ConvNeXt weights, and we reevaluate clean ImageNet in the same run. We report the mean Top-1 over the 75 splits and the mean corruption error (mCE), normalizing the error of each corruption by the corresponding AlexNet error before averaging over corruption types.

Results. The full model reaches a higher mean Top-1 on corrupted images in all ten evaluated pairs (Table S17). In every pair, the corrupted-image gap exceeds the clean gap measured in the same evaluation, and mCE is lower for the full model.

Intermediate readouts under corruption. We also apply, without retraining, the H_0 and H_3 heads of Section A.6 at the end of S_3 and at D_3 to the same 75 splits, on the same images for full and truncated models. Table S18 compares the clean gap and the excess drop, $(\text{clean} - \text{corrupted})_{\text{truncated}} - (\text{clean} - \text{corrupted})_{\text{full}}$, for the native classifier and the H_3 heads. The native excess drop is positive in all ten pairs. At D_3 under H_3, it is negative in eight pairs, close to zero for Swin-B, and positive for Swin-Nano; at the end of S_3, its sign varies. Per-image paired comparisons are available for the eight pairs from Nano-C32 to Small; for the two Base pairs, we report differences of aggregates.

Table S17: Clean ImageNet and ImageNet-C evaluation in FP32. Higher Top-1 (%) and lower mCE are better; clean and corrupted scores use the same checkpoint weights.

	Clean Top-1		ImageNet-C Top-1		mCE	
Model	Full	Truncated	Full	Truncated	Full	Truncated
Swin-T	81.19	78.78	53.17	47.21	59.919	67.479
Swin-T-C64	81.17	80.38	54.52	52.94	58.211	60.275
Swin-S	83.04	82.50	58.00	56.79	53.795	55.329
Swin-B	83.32	83.18	57.93	57.37	53.845	54.603
Swin-Nano-C32	73.84	70.97	41.81	38.04	74.557	79.371
ConvNeXt-T	82.04	80.60	57.34	54.08	54.478	58.542
ConvNeXt-T-C64	81.68	81.31	58.13	56.28	53.469	55.759
ConvNeXt-S	83.24	82.93	61.40	60.28	49.332	50.729
ConvNeXt-B	83.83	83.39	62.16	61.42	48.323	49.283
ConvNeXt-Nano-C32	74.29	71.27	43.91	40.08	71.567	76.388

Table S18: ImageNet-C with the native classifier and with frozen H_3 heads: truncated minus full clean Top-1 and excess drop under corruption (percentage points, mean over the 75 splits). A positive excess drop means that the truncated model loses more under corruption.

	Native		H_3, end of S_3		H_3, D_3	
Model	Clean	Excess drop	Clean	Excess drop	Clean	Excess drop
Swin-Nano-C32	-2.88	$+0.89$	$+14.53$	$+1.44$	$+14.29$	$+1.12$
Swin-T-C64	-0.79	$+0.80$	$+3.66$	-0.91	$+4.93$	-4.46
Swin-T	-2.41	$+3.54$	$+7.02$	$+0.22$	$+8.23$	-1.67
Swin-S	-0.54	$+0.67$	$+1.31$	$+0.29$	$+2.40$	-3.48
Swin-B	-0.14	$+0.42$	-0.01	$+1.64$	$+0.52$	-0.03
ConvNeXt-Nano-C32	-3.02	$+0.81$	$+14.85$	$+1.91$	$+9.60$	-0.51
ConvNeXt-T-C64	-0.37	$+1.48$	$+8.07$	-2.15	$+5.35$	-2.71
ConvNeXt-T	-1.43	$+1.83$	$+10.27$	-3.49	$+7.01$	-3.73
ConvNeXt-S	-0.31	$+0.81$	$+2.23$	-1.04	$+1.94$	-1.13
ConvNeXt-B	-0.44	$+0.30$	$+0.63$	-0.80	$+1.08$	-1.12

A.14 ImageNet-v2 and ImageNet-ReaL: protocol and full results

Protocol. We evaluate all ten Swin and ConvNeXt T/T-C64/S/B/Nano-C32 full/truncated pairs on the 10,000-image matched-frequency ImageNet-v2 test set (Recht et al., 2019) and on ImageNet validation images with ReaL annotations (Beyer et al., 2020). Each variant uses its selected ImageNet checkpoint, with non-EMA weights for ConvNeXt. Inference uses CUDA automatic mixed precision, bicubic resizing of the shorter side to 256, a 224×224 center crop, and ImageNet normalization.

For ReaL, annotations are aligned by validation-image filename; a prediction is correct if any of its Top-k classes matches any reassessed label. The 3,163 images without reassessed labels are excluded, leaving 46,837 images in the denominator for both Top-1 and Top-5. ReaL changes the labels of existing validation images, whereas ImageNet-v2 evaluates newly collected images.

Results. ReaL Top-5 slightly favors the truncated model in all four Small and Base pairs (Table S19).

Table S19: ImageNet-v2 and ImageNet-ReaL accuracy (%) with the selected non-EMA checkpoints.

	ImageNet-v2				ImageNet-ReaL			
	Top-1		Top-5		Top-1		Top-5	
Model	Full	Truncated	Full	Truncated	Full	Truncated	Full	Truncated
Swin-T	70.28	67.50	89.20	87.65	86.67	85.33	97.23	96.71
Swin-T-C64	70.16	68.89	89.42	89.16	86.84	86.29	97.34	97.16
Swin-S	72.37	71.42	90.65	90.15	87.76	87.31	97.54	97.60
Swin-B	72.74	71.92	90.63	90.47	87.82	87.76	97.55	97.66
Swin-Nano-C32	61.65	58.88	83.27	81.49	81.28	78.83	94.97	93.93
ConvNeXt-T	70.96	69.51	89.47	88.77	86.89	86.42	97.26	97.19
ConvNeXt-T-C64	70.32	70.55	89.46	89.18	86.84	86.83	97.29	97.19
ConvNeXt-S	72.72	72.53	90.63	90.75	87.99	87.75	97.71	97.72
ConvNeXt-B	72.93	73.17	90.90	90.60	88.11	87.82	97.60	97.63
ConvNeXt-Nano-C32	62.17	59.50	84.03	81.78	82.04	79.47	95.31	94.08

Table S20: COCO `val2017` AP after 24 epochs of Mask R-CNN training. Subscripts give truncated minus full AP, in points.

Model	Variant	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Swin-S	Full	47.7	69.8	52.3	42.8	66.7	46.0
	Truncated	$46.9_{-0.8}$	$68.9_{-0.9}$	$51.6_{-0.7}$	$42.1_{-0.7}$	$65.9_{-0.8}$	$45.3_{-0.7}$
ConvNeXt-S	Full	46.9	69.7	51.6	42.6	66.4	46.0
	Truncated	$46.0_{-0.9}$	$68.4_{-1.3}$	$50.8_{-0.8}$	$41.7_{-0.9}$	$65.5_{-0.9}$	$44.8_{-1.2}$
Swin-T-C64	Full	45.8	67.9	50.1	41.2	64.7	44.1
	Truncated	$45.2_{-0.6}$	$67.1_{-0.8}$	$49.6_{-0.5}$	$40.7_{-0.5}$	$64.2_{-0.5}$	$43.5_{-0.6}$
ConvNeXt-T-C64	Full	45.6	67.4	49.9	41.2	64.6	44.0
	Truncated	$45.3_{-0.3}$	$67.2_{-0.2}$	$50.0_{+0.1}$	$40.7_{-0.5}$	$64.2_{-0.4}$	$43.6_{-0.4}$

A.15 COCO DETECTION AND INSTANCE SEGMENTATION

We transfer the Tiny-C64 and Small full/truncated pairs of Swin and ConvNeXt to COCO `train2017` with a standard Mask R-CNN and evaluate on `val2017`. All variants use the same FPN and detection heads. The FPN receives (S_1, S_2, S_3, S_4) from a full backbone and (S_1, S_2, S_3, D_3) from a truncated one, with strides $(4, 8, 16, 32)$. We initialize from the selected ImageNet checkpoints, using non-EMA weights for ConvNeXt. Training uses AdamW, a global batch size of 16, multi-scale inputs with a short side between 480 and 800 pixels, and the family-specific learning rates and layer decay of our detection configurations. The schedule runs for 24 epochs, with learning-rate drops at epochs 18 and 22.

Table S20 reports the validation metrics after the complete 24-epoch schedule. Across the four pairs, box AP falls by 0.3–0.9 points and mask AP by 0.5–0.9. All six metrics decrease except box AP at IoU 0.75 for ConvNeXt-T-C64, which increases by 0.1 points.

A.16 DETAILED MEASUREMENT DEFINITIONS

For a feature map $Z \in \mathbb{R}^{L\times C}$, let $P(Z) = L^{-1}\sum_{\ell=1}^{L} Z_\ell$ denote global average pooling (Lin et al., 2013). The native readout can be written as

$$H_\phi(Z) = A_\phi(\mathcal{N}_{\mathrm{post}}(P(\mathcal{N}_{\mathrm{pre}}(Z)))), \tag{S3}$$

where A_ϕ is an affine classifier and $\mathcal{N}_{\mathrm{pre}}$ and $\mathcal{N}_{\mathrm{post}}$ denote the normalizations that a family applies before and after pooling; either may be the identity. The plain symbol N is reserved for block

counts (Section A.3). This notation preserves the native normalization type, including its spatial dependence when present.

Removal with fixed weights. Given a full-model checkpoint, whose learned parameters we mark with a star, we evaluate Equation 1 of the main paper with either $T = S_{4,\psi^\star}$ or $T = \mathrm{Id}$, keeping all surviving weights fixed. For an evaluation set $\mathcal{V}$, the measured difference is

$$\Delta_{\mathrm{fixed}} = \mathrm{Acc}_{\mathcal{V}}(H_{\phi^\star} \circ S_{4,\psi^\star} \circ D_3^\star \circ E_{\theta^\star}) - \mathrm{Acc}_{\mathcal{V}}(H_{\phi^\star} \circ D_3^\star \circ E_{\theta^\star})\,. \tag{S4}$$

This intervention measures what removal costs under the representation and the readout that the full model learned.

Similarity between trained representations. We compare full and truncated checkpoints at matching attachment points using linear centered kernel alignment (CKA) (Kornblith et al., 2019). Let X and Y contain their GAP features for the same images, with columns centered across images. We use

$$\mathrm{CKA}(X, Y) = \frac{\|X^\mathsf{T} Y\|_F^2}{\|X^\mathsf{T} X\|_F \|Y^\mathsf{T} Y\|_F}. \tag{S5}$$

This measures the similarity of the pooled representations at the end of training and does not involve the fitted probe heads.

Effective receptive field. We characterize spatial input sensitivity with a gradient-based effective receptive field (ERF), following the use of input gradients to measure receptive-field extent (Luo et al., 2016). At attachment point s, let $Z_s(x)_{q_s} \in \mathbb{R}^{C_s}$ be the feature vector at the probe unit q_s, taken at index $\lfloor n_s/2 \rfloor$ along each axis of the $n_s \times n_s$ grid. That index is the median unit on an odd grid but the first unit past the middle on an even one, so the probe is not exactly grid-centered at every attachment point. For each image and random sign vector $v \in \{-1, +1\}^{C_s}$, define $a_{s,v}(x) = C_s^{-1} v^\mathsf{T} Z_s(x)_{q_s}$. The ERF map is

$$M_s(p) = \left[\mathbb{E}_{x,v} \frac{1}{3} \sum_{c=1}^{3} \left(\frac{\partial a_{s,v}(x)}{\partial x_{p,c}} \right)^2 \right]^{1/2}, \tag{S6}$$

where p indexes positions of the normalized input tensor the model receives and c indexes its three channels. The expectation is estimated by averaging over the evaluated images and sign projections before the square root is taken. Radial measurements are centered on the input location that corresponds to q_s. That location is set by the model's own reduction chain and is not in general the center of the image, so map mass is normalized within the largest disk that is centered on it and contained in the image. The reported r_{50} is the smallest radius that encloses half of the mass of that disk. We compare S_3 across architectures, together with the terminal outputs: S_4 for the full model and D_3 for the truncated model (Section 4.2 of the main paper).

A.17 Terminal readouts and spatial-mixing controls

Token-wise nonlinear readout. We compare the placement of an MLP relative to pooling on frozen D_3 features. Let $\bar{Z}$ denote the feature map after the normalization shared by the two conditions. With affine maps $A : \mathbb{R}^C \to \mathbb{R}^{4C}$ and $B : \mathbb{R}^{4C} \to \mathbb{R}^{1000}$, and $\sigma = \mathrm{GELU}$ (Hendrycks & Gimpel, 2016), the two readouts are

$$H_{\mathrm{post}}(Z) = B\big(\sigma(A(P(\bar{Z})))\big)\,, \qquad H_{\mathrm{pre}}(Z) = B\big(P(\sigma(A(\bar{Z})))\big)\,. \tag{S7}$$

In the second expression, A and σ act independently at each position with shared parameters. Pooling commutes with the affine map A, but in general it does not commute with σ. Both readouts have the same number of parameters; the pre-pooling version only evaluates the first projection at every position. For Swin, $\bar{Z}$ uses the native per-position LayerNorm, and for ConvNeXt, $\bar{Z} = Z$ in both conditions. The paired heads are trained separately from identical initial weights.

Spatial mixing control. For the matched Mixer/channel-only comparison, write one Mixer block as

$$\begin{aligned} U &= Z + \mathcal{D}_1(M_{\mathrm{tok}}(\mathrm{LN}_1(Z)))\,, \\ T_{\mathrm{block}}(Z) &= U + \mathcal{D}_2(M_{\mathrm{ch}}(\mathrm{LN}_2(U)))\,. \end{aligned} \tag{S8}$$

Here M_{tok} is an MLP over the L positions, applied with weights shared across channels, and M_{ch} is an MLP over channels, applied with weights shared across positions. $\mathcal{D}_1$ and $\mathcal{D}_2$ denote stochastic-depth applications (Huang et al., 2016). Per-position LayerNorm uses $\epsilon = 10^{-5}$. We stack two such blocks on the 7×7 terminal grid. The channel-only control replaces M_{tok} by the identity, so its first branch remains $U = Z + \mathcal{D}_1(\mathrm{LN}_1(Z))$, and it keeps both normalizations, both residual branches, and both stochastic-depth applications. For the base-width-64 Swin comparison, the channel hidden width increases from 256 to 261 to approximately match the parameter budget once the token MLP is removed; the token hidden width in the Mixer is 49. The contrast tests learned spatial mixing at nearly equal parameter counts, with a small compensating change in channel-MLP width and unchanged branch structure.

Token-wise terminal stage of the short factorial. The token-wise stage C of Section A.9 follows the native Swin block. The spatial stage S uses width $d = 256$ and eight attention heads per block. Writing A_{attn} for the window-attention branch of an S_4 block, C keeps

$$U = Z + \mathrm{Lin}_{d\times d}(\mathrm{LN}_1(Z)), \qquad T_{\mathrm{block}}(Z) = U + M_{5.5d}(\mathrm{LN}_2(U)), \tag{S9}$$

where $\mathrm{Lin}_{d\times d}$ is an affine map replacing A_{attn} and $M_{5.5d}$ is the native MLP widened from $4d$ to $5.5d$. A single affine map replaces the value and output projections, whose composition spans the same affine functions with twice as many parameters. The two blocks of C therefore mix channels at each position without mixing positions. Each C block has 789,376 parameters, 0.22% fewer than the 791,112 of an S block including its relative-position table.

A.18 Extended discussion

The contribution of S_4 depends on the intervention. Fixed-weight removal, frozen-feature probing, and training without S_4 answer different questions. On Swin-T, direct removal lowers Top-1 from 81.17 to 58.43, fitting a fresh native head at D_3 reaches 70.20 (Section A.1), and training the truncated architecture from scratch reaches 78.78, against 81.20 for its locally trained full reference (Table 1 of the main paper). These numbers suggest that the large loss after direct removal reflects both how much the readout depends on S_4 features and what the earlier stages learned while S_4 was present. Because the three protocols use different optimization settings and, in the from-scratch case, different backbone checkpoints, their gaps cannot be added into a decomposition of the contribution of S_4. The sensitivity of the probes to normalization also shows that probe accuracy measures accessibility under a specified readout, not all the task-relevant information that a feature map holds. Finally, the eight D_3 probe entries that peak in the final two head-training epochs may understate converged scores, including the Swin-T H_1 gap and the MogaNet-S normalization contrast.

Adaptation changes the organization of the surviving hierarchy. The representation measurements support the view that the role S_3 learns depends on the architecture downstream of it. They do not establish that S_3 reproduces the computation of S_4, nor that an ERF change causes accuracy to recover: CKA compares pooled features of independently trained networks, and the ERF measures input sensitivity, not task-specific use of context. The pooled-feature geometry statistics are likewise descriptive and lack calibrated intervals. The exploratory spectrum is also descriptive: the 7×7 grid provides few frequency bins, and its changes do not track the S_3 ERF differences uniformly.

Readout parameterization. The normalization contrasts inside D_3 do not establish that normalization placement explains the difference between families. Repeating the global average four times illustrates the sensitivity of H_0 to parameterization: with AdamW and no weight decay, the four identical blocks receive identical updates, multiplying the effective step size by four. We therefore do not interpret H_0 differences of this size between views of different widths as differences in the information available to a linear classifier.

Terminal computation includes distinct operations. The pre-pooling MLP operates independently at each spatial position. Its gains are consistent with nonlinear processing before aggregation, subject to the cache-precision confound, but do not establish interactions between positions (Table S16). The separate token-and-channel Mixer comparison tests terminal spatial mixing (Table 3 of the main paper).

Interpreting corruption readouts. The ImageNet-C comparisons do not separate robustness specific to S_4 from the effects of reduced capacity and lower clean accuracy, and frozen heads at S_3 and D_3 do not uniformly reproduce the native excess drop (Table S18). Because the truncated heads start from higher clean accuracy and the full D_3 heads exclude the native terminal computation, these differences do not localize the native robustness gap. Evaluating the reallocated Tiny models and the lightweight variants on corruptions would show whether their recovery in clean accuracy carries over.

Readout parity, spatial restrictions, and early learning. The small readout gains at Base are consistent with the small from-scratch truncation costs but do not make the trained terminal stage dispensable: bypassing S_4 in the locally trained Base models still costs 0.25 points for Swin and 0.52 for ConvNeXt. The Nano allocation controls do not factorially separate depth, capacity, and stage position. Fixed-weight restrictions take the networks out of their training distribution and measure dependence, not necessity; because restrictions and bypass change the network in different ways, their accuracy costs are not additive parts of one mechanism. The short factorial supports complementarity during early learning of one Swin-Nano configuration, inside the warmup, but it contains no model without S_4 and does not test whether the final truncated representation reduces the need for terminal spatial mixing. A positive interaction in both seeds does not give a precise estimate of its variance. The ordinary training curves motivate this factorial but do not identify the cause of their early gap, since the variants also differ in parameters, depth, FLOPs, or stochastic depth.