RADIOv2.5: Improved Baselines for Agglomerative Vision Foundation Models

Supplementary Material

Variant	Zero Shot	kNN	ADE20k	Depth	SNorm	MultiView	SPairs
Stage 1							
No SAM	**79.38**	83.17	50.27	82.54	61.03	58.12	51.97
SAM	79.37	**83.29**	**51.14**	**82.60**	**61.88**	**58.91**	**52.49**
Stage 2							
No SAM	80.43	83.83	50.24	**83.29**	61.43	58.98	**54.72**
SAM	**80.47**	**83.92**	**51.36**	83.17	**62.80**	**61.36**	54.66

Table 6. Ablation on whether to include SAM in the teacher set for the first two stages of multi-stage training.

#	Zero Shot	kNN	ADE20k	Depth	SNorm	MultiView	SPairs	SAM
Stage 1								
1	**79.37**	**83.29**	**51.14**	82.60	**61.88**	**58.91**	52.49	**71.51**
2	78.70	83.00	50.88	82.32	60.78	58.49	52.32	71.29
4	77.93	82.61	51.09	**82.89**	61.26	58.47	**52.54**	71.02
Stage 2								
1	**80.47**	**83.92**	**51.36**	**83.17**	**62.80**	**61.36**	54.66	**73.62**
4	78.82	83.41	51.34	82.99	61.58	59.80	**55.96**	73.03

Table 7. Ablation on number of partitions for the first two stages of multi-stage training.

A. Additional Experimental Findings

A.1. Is SAM a good teacher?

SAM [22] has been a controversial choice in the recent agglomerative models literature. AM-RADIO [35] struggled to prove that its inclusion improved any metrics. Theia [38] specifically ablated whether to include SAM, and found that it degraded their metrics. UNIC [37] opted as well not to include SAM. Based on the findings with PHI-S [34], and our confirmation of imbalance in section 4.4, it seems that a major problem with SAM may just have been that interpolating its features is a "really bad thing", and also that the distribution was extremely unbalanced; something that PHI-S corrects. We chose to re-run the study of whether SAM is a good teacher now that we have a new bag of tricks. In particular, we run the first two stages of multi-stage training, we use the mosaic augmentation for SAM in both of these stages, and we apply PHI-S to all teachers. We show the results in table 6 where it is clear that including SAM has negligible (but positive) impact on our classification benchmarks, and strong positive effects on dense tasks such as semantic segmentation and 3D probing. SAM's inclusion also enables novel opportunities such as those found in VideoSAM [17] where they employ AM-RADIO's SAM adaptor and backbone simultaneously for video segmentation.

Finding 6. All teachers are beneficial, including SAM, despite recent trends. It also has broad downstream applicability, granting our student the same abilities.

A.2. Partitioning

In PHI-S [34], the authors opted to put teachers in their own partitions, which reduces the teacher overhead (as the per-teacher batch size is reduced). However, the paper does not ablate whether this choice came with model quality consequences. In Table 7 we study the number of partitions for the first two stages of training. We find that fewer partitions is strongly better for summarization tasks, and less clear for dense tasks.

Finding 7. Minimizing the number of partitions seems to be beneficial, assuming you can afford the teacher overhead. Under compute constraints, partitioning is an effective strategy to reduce the overhead.

A.3. SigLIP Teacher

Building on previous work ([42], [23]), we replace OpenAI-CLIP [32] with SigLIP [49], defining this as our configuration $\mathcal{C}$. Our choice is validated by the significant improvements observed in VLM tasks, as shown in Table 1.

Layers	Aggregation	Head	ADE20k	Depth	Surf Normals	Overall
31	N/A	Linear	52.47	82.9	57.0	61.215
7-15-23-31	Sparse	Linear	52.99	82.5	59.6	62.03
(0-9)-(10-19)-(20-30)-31	Dense	Linear	52.96	82.7	59.5	62.03
15-31	Sparse	Linear	52.90	83.1	59.6	62.12
(0-15)-(16-30)-31	Dense	DPT	54.27	85.4	60.7	63.65
15-31	Sparse	DPT	54.58	84.6	61.0	63.70
7-15-23-31	Sparse	DPT	**55.19**	85.9	61.6	64.46
(0-9)-(10-19)-(20-30)-31	Dense	DPT	54.28	85.5	62.3	64.08
3-7-11-15-19-23-27-31	Sparse	DPT	54.42	**86.7**	**62.8**	**64.58**
		TextVQA	**ChartQA**	**DocVQA**	**InfoVQA**	**OCRBench**
Last	N/A	**63.6**	23.4	47.0	33.8	**42.0**
7-15-23-31	Sparse	63.2	**24.1**	**47.2**	**34.3**	40.3
(0-9)-(10-19)-(20-30)-31	Dense	63.5	23.1	47.0	33.5	40.2

Table 8. Study on the effect of RADIOv2.5-H feature selection. For "dense" aggregation, the numbers in brackets indicate the range of layers from which the average is calculated. **Top:** Semantic segmentation, depth estimation, surface normals estimation. **Bottom**: VILA benchmarks. Pixel-level tasks exhibit a clear preference for more layers and non-linear heads, while VLM tasks seem mostly neutral to this choice.

A.4. Feature Selection

For each image, our foundation model outputs a summary vector along with patch tokens at a granularity of one per 16^2 input pixel block. For image-level tasks such as classification, search, or curation, the summary vector provides a rich embedding. For dense tasks, such as segmentation or 3D understanding, the patch tokens are a natural choice. However, as demonstrated in previous work, incorporating additional intermediate activations further enhances performance. For example, [12] uses a Dense Prediction Transformer (DPT) head [33] for 3D reasoning, while [48] averages multiple ranges of intermediate activations to feed into an LLM for VLMs.

In this section, we investigate various feature selection methods and present the results in Table 8. We experiment with *sparse* feature selection (selecting activations from individual layers throughout the model) and *dense* feature selection (aggregating information across all layers by averaging groups of layer activations). We examine the impact of feature selection in conjunction with different downstream heads (linear or DPT probe).

Our findings show that a linear probe alone is insufficient to leverage additional information from intermediate layer activations. However, when a DPT head is employed, it effectively incorporates this additional information. We note, however, that it is challenging to disentangle the benefits provided by additional feature information, and those of the extra learnable parameters of the DPT head. Unlike [48], we do not observe a positive impact of using dense features in VLMs.

Finding 8. Intermediate layer activations greatly benefit downstream tasks if a non-linear transformation is employed.

B. VLM Benchmarks

B.1. More Vision Encoder Comparisons

In Table 9, we report benchmark results for OpenAI-CLIP-336, AM-RADIO-H, SigLIP-400m, and RADIOv2.5-H. The same Qwen2-7B-Instruct LLM, training data, and hyperparameters are used across all configurations. RADIOv2.5-H is utilized in conjunction with Token Merging and configured to produce either 196 tokens per image (matching SigLIP) or 512 tokens per image (to demonstrate scaling capabilities). The results show a significant improvement when transitioning to RADIOv2.5-H, even with the same token count as the SigLIP baseline. Furthermore, additional benefits are observed when scaling up the number of vision tokens.

B.2. Scaling up to More Data

In Table 10, we report benchmark results obtained using an improved SFT data mixture, which includes data from ShareGPT4v, LLaVA Instruct, Cambrian, VFLAN, and the training sets of some benchmarks, for a total of 9.8M samples.

Vision Encoder	Resolution	Tokens/im	TextVQA	ChartQA	DocVQA	InfoVQA	OCRBench	GQA	POPE	MME	SEED(I)	AI2D	Average
OpenAI-CLIP [32]	336^2	144	63.8	27.5	48.8	33.0	414	63.0	86.7	1646.9	66.7	67.1	58.03
AM-RADIO-H [35]	512^2	256	55.9	15.7	35.2	30.6	316	60.2	85.3	1516.5	71.8	63.1	52.52
SigLIP-SO400M [49]	384^2	196	67.6	33.0	57.1	36.0	458	63.0	85.7	1605.2	74.2	**68.6**	61.13
RADIOv2.5-H (ours)	768^2	196	**70.8**	33.9	59.5	37.0	482	63.9	86.9	**1613.5**	75.1	66.7	62.27
RADIOv2.5-H (ours)	768^2	512	70.2	**37.3**	**64.2**	**37.6**	**523**	**64.5**	**87.3**	1587.6	**75.4**	67.5	**63.57**

Table 9. VILA benchmark results for various vision encoders. We used **Qwen2-7B-Instruct** as LLM and **ShareGPT4v[6] and VFLAN[45]** data. From left to right we report: image resolution, numbers of tokens per image, TextVQA (with OCR hints) validation accuracy, ChartQA overall, DocVQA validation accuracy, InfoVQA validation accuracy, OCRBench accuracy, GQA (TestDev) accuracy, POPE F1 score, MME perception score, SEED (Image) accuracy, AI2D accuracy, average (calculated after dividing MME score by 20 and OCR score by 10).

Vision Encoder	TextVQA	ChartQA	DocVQA	InfoVQA	OCRBench	GQA	POPE	MME	SEED(I)	AI2D	Average
SigLIP-SO400M [49]	69.7	67.2	63.7	40.9	588	62.4	86.9	1648.5	72.0	**75.0**	67.90
RADIOv2.5-H (ours)	**71.5**	**71.9**	**73.6**	**45.6**	**667**	**62.7**	**87.5**	**1664.1**	**77.39**	74.8	**71.49**

Table 10. VILA benchmark results for SigLIP and RADIOv2.5-H. We used **Qwen2-7B-Instruct [44]** as LLM, with an improved data mixture of public datasets. From left to right we report: TextVQA (with OCR hints) validation accuracy, ChartQA overall, DocVQA validation accuracy, InfoVQA validation accuracy, OCRBench accuracy, GQA (TestDev) accuracy, POPE F1 score, MME perception score, SEED (Image) accuracy, AI2D accuracy, average (calculated after dividing MME score by 20 and OCR score by 10).

RADIOv2.5-H is used in conjunction with Token Merging and is configured to produce 512 tokens per image. RADIOv2.5-H outperforms the SigLIP baseline on all benchmarks, except for AI2D, where it achieves a tie with SigLIP.

B.3. Effect of the Compression Method

Vision Encoder	Compression	Tokens/im	TextVQA	ChartQA	DocVQA	InfoVQA	OCRBench	GQA	POPE	MME	SEED(I)	AI2D	Average
SigLIP SO400M [49]	2×2 Unshuffle	196	**63.4**	**23.1**	41.4	**30.1**	**339**	63.8	85.4	1518.4	70.9	63.6	**55.15**
SigLIP SO400M [49]	ToMe r=533	196	62.9	21.2	**41.8**	**30.1**	326	**64.2**	**85.8**	**1537.5**	**71.7**	**63.7**	55.09
RADIOv2.5-H (ours)	2×2 Unshuffle	576	66.4	25.0	53.3	32.5	402	**64.8**	86.5	1434.0	73.4	63.9	57.77
RADIOv2.5-H (ours)	ToMe r=2048	256	**69.7**	30.4	52.3	36.2	429	63.8	86.8	**1572.4**	**74.6**	65.1	60.04
RADIOv2.5-H (ours)	ToMe r=3584	512	68.9	**31.3**	**53.9**	**37.0**	**446**	63.0	**87.6**	1537.7	73.9	**66.0**	**60.31**

Table 11. VILA benchmark results for various vision encoders and compression methods. We used **MN-Minitron-8B** as LLM and **ShareGPT4v[6] and VFLAN[45]** data. From left to right we report: number of vision tokens per image, TextVQA (with OCR hints) validation accuracy, ChartQA overall, DocVQA validation accuracy, InfoVQA validation accuracy, OCRBench accuracy, GQA (TestDev) accuracy, POPE F1 score, MME perception score, SEED (Image) accuracy, AI2D accuracy, average (calculated after dividing MME score by 20 and OCR score by 10).

In Table 11, we report benchmark results for SigLIP-400m and RADIOv2.5-H using different token compression methods. The same MN-Minitron-8B LLM, training data, and hyperparameters are used across all configurations. The results show no improvement when applying Token Merging to SigLIP. RADIOv2.5-H, on the other hand, demonstrates significant improvement with Token Merging, and increasing the token count from 256 to 512 provides a modest additional gain.

B.4. Block-wise vs Output Token Merging

In this section we evaluate which works better: merging tokens incrementally in each ViT block (using "keys" as criteria as in the original ToMe [4] formulation), or merging tokens once at the output of the ViT. Results are shown in Table 12 and indicate that in our setup, one-shot merging at the output yields improved results.

Token Merging	Resolution	Tokens/im	TextVQA	ChartQA	DocVQA	InfoVQA	OCRBench	GQA	POPE	MME	SEED(I)	MMMU	AI2D	Average
Block-wise ($r =\sim 65$)	768^2	196	65.2	61.3	52.4	29.3	469	63.4	**87.7**	1539.4	73.9	42.8	79.5	61.76
Output ($r = 2108$)	768^2	196	**68.8**	**61.6**	**54.2**	**30.8**	**498**	**63.8**	87.3	**1562.3**	**74.1**	**44.1**	**82.5**	**63.19**

Table 12. Comparison of block-wise vs output token merging. We use a RADIOv2.5-H vision encoder. **"Block-wise" token merging**: we merge tokens in each successive ViT block, using keys as criteria. We assign 50% of tokens as target tokens and set $r = 65$ for the first 31 blocks and $r = 93$ for the last block, totaling 2108 merged tokens and bringing the final number of tokens to 196. **"Output" token merging**: we only merge the ViT output tokens, using token values as criteria. We partition source/targets tokens in a 6×6 strided pattern and set r=2108.

B.5. High-Resolution Inference using Tiling

Vision Encoder	Resolution	Compression	Tokens/im	TextVQA	ChartQA	DocVQA	InfoVQA	OCRBench	GQA	POPE	MME	SEED(I)	AI2D	Average
SigLIP SO400M [49] + Tiling	Up to 13×384^2	2×2 Unshuffle	~ 1928	66.5	**64.6**	**74.4**	40.1	521	**64.5**	**88.0**	1536.1	74.4	79.3	68.07
RADIOv2.5-H (ours)	768^2	ToMe r=2108	196	68.8	61.4	61.9	36.7	498	63.8	88.4	1562.6	74.1	71.9	65.49
RADIOv2.5-H (ours) + Tiling	Up to 7×768^2	ToMe r=2108	~ 1233	**73.2**	63.5	73.1	**46.5**	**545**	**64.5**	87.9	**1611.6**	**75.3**	**82.4**	**70.15**

Table 13. VILA benchmark results using tiling to emulate high-resolution inference. We used **MN-Minitron-8B** as LLM and the data mixture from **LLaVA1.6**. From left to right we report: number of vision tokens per image (calculated across all benchmarks), TextVQA (with OCR hints) validation accuracy, ChartQA overall, DocVQA validation accuracy, InfoVQA validation accuracy, OCRBench accuracy, GQA (TestDev) accuracy, POPE F1 score, MME perception score, SEED (Image) accuracy, AI2D (No Mask) accuracy, average (calculated after dividing MME score by 20 and OCR score by 10).

In Table 13, we emulate high-resolution inference using tiling [7]. While we acknowledge that there is no strict equivalence in the number of pixels (SigLIP processes an average of 1.8M pixels per sample, whereas RADIOv2.5-H processes an average of 3.9M pixels) or in the number of generated tokens (SigLIP outputs an average of 2,410 tokens per sample, compared to 1,313 tokens for RADIOv2.5), we observe slightly improved accuracy with RADIOv2.5 despite producing a significantly smaller number of output tokens.

C. Semantic Segmentation

Model	Params	ADE20k			Pascal VOC		
		512	768	1024	512	768	1024
RADIOv2.5-B (ours)	98M	**48.94**	**50.48**	**51.16**	**84.35**	**85.47**	**85.33**
AM-RADIO-L [35]*	320M	50.03	37.99	35.63	83.76	68.85	63.86
+ multi-res	320M	**51.54**	51.74	52.84	**85.51**	86.84	**87.21**
RADIOv2.5-L (ours)	320M	51.47	**51.90**	**52.95**	85.49	**86.96**	87.03
AM-RADIO-H [35]*	553M	51.34	35.78	32.99	84.71	64.54	59.15
RADIOv2.5-H (ours)	553M	**51.58**	**52.45**	**53.91**	**85.97**	**87.54**	**87.69**
DINOv2-g-reg	1.1B	48.79	48.37	50.71	82.72	83.95	84.25

Table 14. Semantic segmentation mIoU for ADE20k and Pascal VOC across different models and resolutions: 512×512, 768×768, and 1024×1024. Note: For DINOv2, we use the nearest larger multiple of its patch size (14). We apply a linear probe on top of the frozen features from the vision encoder. The mIoU of RADIOv2.5-B exceeds that of DINOv2-g-reg, despite being only one-tenth the size. *Our reproduction. Mode switching in AM-RADIO is also evident when resolution exceeds 512, as the mIoU sharply drops.

In Table 14, we report Semantic Segmentation mIoU at different resolutions on ADE20k [50] and VOC [14]. We observe

that RADIOv2.5 favorably scales to higher resolutions, while the accuracy of AM-RADIO falls above a resolution of 512×512.

D. Additional Benchmarks

Following MLoRE [47] we report metrics on NYUDv2 and PASCAL Context for our model, along with the other agglomerative models (Theia [38] and UNIC [37]), as well as DINOv2 [9] as it's the core perception teacher for these types of tasks. Previously to this study, MLoRE was the state-of-the-art for producing a multi-task model for these metrics. We keep the backbone model frozen, and use MLoRE's "conv" head for each task. The conv head is defined as follows:

$$\text{Conv-3x3} \rightarrow \text{BatchNorm} \rightarrow \text{GeLU} \rightarrow \text{Conv-1x1} \tag{6}$$

At the largest scale, RADIOv2.5-H is extremely competitive with DINOv2-g at half the parameters. At the ViT-B and ViT-L scale, RADIOv2.5 is the closest to DINOv2 of the same size of any of the current agglomerative models. We train with a learning rate of $1e-3$, and use a weight of 1 for all tasks. We purposefully don't tune any hyperparameters.

Model	Backbone	SemSeg mIoU ↑	Depth RMSE ↓	Normal mErr ↓	Boundary Loss ↓
MLoRE	Custom	55.96	0.5076	18.33	-
DINOv2	ViT-B/16	**60.64**	**0.4816**	**18.19**	0.1268
Theia	ViT-B/16	38.90	0.6377	24.11	0.1298
UNIC	ViT-B/16	42.21	0.6172	22.78	0.1285
RADIOv2.5	ViT-B/16	57.19	0.4980	20.04	**0.1263**
DINOv2	ViT-L/14	**62.94**	**0.4406**	**17.63**	0.1266
UNIC	ViT-L/14	58.56	0.4916	19.34	0.1274
RADIOv2.5	ViT-L/16	61.42	0.4577	18.57	**0.1259**
AM-RADIO	ViT-H/16	62.76	**0.4339**	18.43	0.1266
RADIOv2.5	ViT-H/16	**63.82**	0.4353	**17.67**	**0.1256**
DINOv2	ViT-g/14	63.89	0.4252	17.20	0.1262

Table 15. Multi-task dense results on the NYUDv2 dataset. Note that we report the "Boundary Loss" and not the osdF metric due to the latter having a dependency on Matlab to compute.

Model	Backbone	SemSeg mIoU ↑	Parsing mIoU ↑	Saliency maxF ↑	Normal mErr ↓	Boundary Loss ↓
MLoRE	Custom	81.41	70.52	**84.90**	**13.51**	-
DINOv2	ViT-B/16	81.68	**73.24**	77.54	17.44	0.0619
Theia	ViT-B/16	69.84	60.67	80.63	16.94	0.0623
UNIC	ViT-B/16	75.90	62.85	**81.84**	**15.78**	0.0620
RADIOv2.5	ViT-B/16	**81.75**	71.49	81.26	16.10	**0.0618**
DINOv2	ViT-L/14	81.97	**74.51**	77.22	17.77	0.0620
UNIC	ViT-L/14	81.82	72.24	79.21	17.35	0.0621
RADIOv2.5	ViT-L/16	**82.87**	74.32	81.65	16.15	**0.0617**
AM-RADIO	ViT-H/16	82.78	74.42	78.48	17.53	0.0619
RADIOv2.5	ViT-H/16	**83.43**	**75.75**	81.19	16.16	**0.0617**
DINOv2	ViT-g/14	82.47	75.56	76.93	17.59	0.0619

Table 16. Multi-task dense results on the PASCAL Context dataset. Note that we report the "Boundary Loss" and not the osdF metric due to the latter having a dependency on Matlab to compute. We also don't bold a cell in a model size group if a smaller model has even higher quality for a given benchmark.

E. Hyperparameters

In Table 17, we report our RADIOv2.5 training parameters.

Parameter	Value	
	RADIOv2.5-L	RADIOv2.5-H
Backbone	ViT-L	ViT-H
Learning Rate	$1e^{-3}$	
Weight Decay	$2e^{-2}$	
Teachers	DFN CLIP SigLIP 400m DINOv2-g-reg SAM-H	
Feature Normalization	PHI-S	
Dataset	DataComp-1B	
Feature Distillation Loss	MSE	
Summary Loss	Cosine	
Backbone pre-training	ImageNet-1k	

Table 17. RADIOv2.5 Training Hyperparameters

F. A Measure of Scale Equivariance

We define a measure of scale equivariance σ^2_{scale}: given an array of feature tensors $\{F_i\}$ with shapes (H_i, W_i, C):

1. Let F_{min} denote the tensor with the smallest spatial dimensions $(H_{\text{min}}, W_{\text{min}}, C)$.
2. Compute the per-channel mean and variance of F_{min}:

$$\mu_c = \frac{1}{H_{\text{min}} W_{\text{min}}} \sum_{h=1}^{H_{\text{min}}} \sum_{w=1}^{W_{\text{min}}} F_{\text{min}}(h, w, c) \tag{7}$$

$$\sigma_c^2 = \frac{1}{H_{\text{min}} W_{\text{min}}} \sum_{h=1}^{H_{\text{min}}} \sum_{w=1}^{W_{\text{min}}} \left(F_{\text{min}}(h, w, c) - \mu_c\right)^2 \tag{8}$$

3. Normalize each tensor F_i using μ_c and σ_c:

$$\hat{F}_i(h, w, c) = \frac{F_i(h, w, c) - \mu_c}{\sigma_c} \tag{9}$$

4. Bilinearly interpolate each normalized tensor $\hat{F}_i$ down to $(H_{\text{min}}, W_{\text{min}}, C)$, resulting in tensors $\{\tilde{F}_i\}$.
5. Stack all resized tensors $\{\tilde{F}_i\}$ along a new dimension and compute variance along this new dimension:

$$\sigma^2(h, w, c) = \text{Var}(\{\tilde{F}_i(h, w, c)\}) \tag{10}$$

6. Finally, compute the average variance over the spatial dimensions:

$$\sigma^2_{\text{scale}} = \frac{1}{H_{\text{min}} W_{\text{min}}} \sum_{h=1}^{H_{\text{min}}} \sum_{w=1}^{W_{\text{min}}} \sigma^2(h, w, c) \tag{11}$$

F.1. Scale Variance Implementation

```
1
2  def scale_variance(tensors: List, scale_up: bool):
3      """Compute feature variance across scales.
4
5      Steps:
6      * Per-channel standardization using the stats (mean/std)
7        of largest features (if scale_up) or smallest features (if scale_down)
```

```
    * Interpolation to the size of the largest features (if scale_up) or
      smallest features (if scale_down).
    * Stack along a new dimension.
    * Compute variance along the new dimension.
    * Average across batch and spatial dimensions.
    """

    if scale_up:
        target_tensor = max(tensors, key=lambda x: x.numel())
        # Find the largest spatial dimensions
        target_H = max(tensor.shape[1] for tensor in tensors)
        target_W = max(tensor.shape[2] for tensor in tensors)
    else:
        target_tensor = min(tensors, key=lambda x: x.numel())
        # Find the smallest spatial dimensions
        target_H = min(tensor.shape[1] for tensor in tensors)
        target_W = min(tensor.shape[2] for tensor in tensors)

    # Compute mean and std along spatial dimensions (H, W) for each channel
    mean = target_tensor.mean(dim=(1, 2), keepdim=True)
    std = target_tensor.std(dim=(1, 2), keepdim=True)

    # Normalize each tensor and resize to the largest spatial dimensions
    normalized_tensors = []
    for tensor in tensors:
        # Mean-center and normalize
        normalized_tensor = (tensor - mean) / (std + 1e-8)  # Adding a small epsilon to
            avoid division by zero

        # Resize to (B, max_H, max_W, C)
        resized_tensor = F.interpolate(
            normalized_tensor.permute(0, 3, 1, 2),
            size=(target_H, target_W),
            mode='bilinear',
            align_corners=False
        ).permute(0, 2, 3, 1)

        normalized_tensors.append(resized_tensor)

    stacked_tensors = torch.stack(normalized_tensors)  # Shape: (num_tensors, B, max_H,
        max_W, C)

    # Compute variance along the first dimension (num_tensors)
    variance_tensor = torch.var(stacked_tensors, dim=0)

    return variance_tensor.mean().item()
```

G. Mode-Switch PCA Visualizations

Figure 9 shows more visualization of the RADIO feature changes incurred by resolution increases.

H. Visualizations of Native vs Emulated High-Resolution Inference

Figure 10 shows visualizations of output features for emulated high-resolution inference through tiling, and for native high-resolution inference using RADIOv2.5.

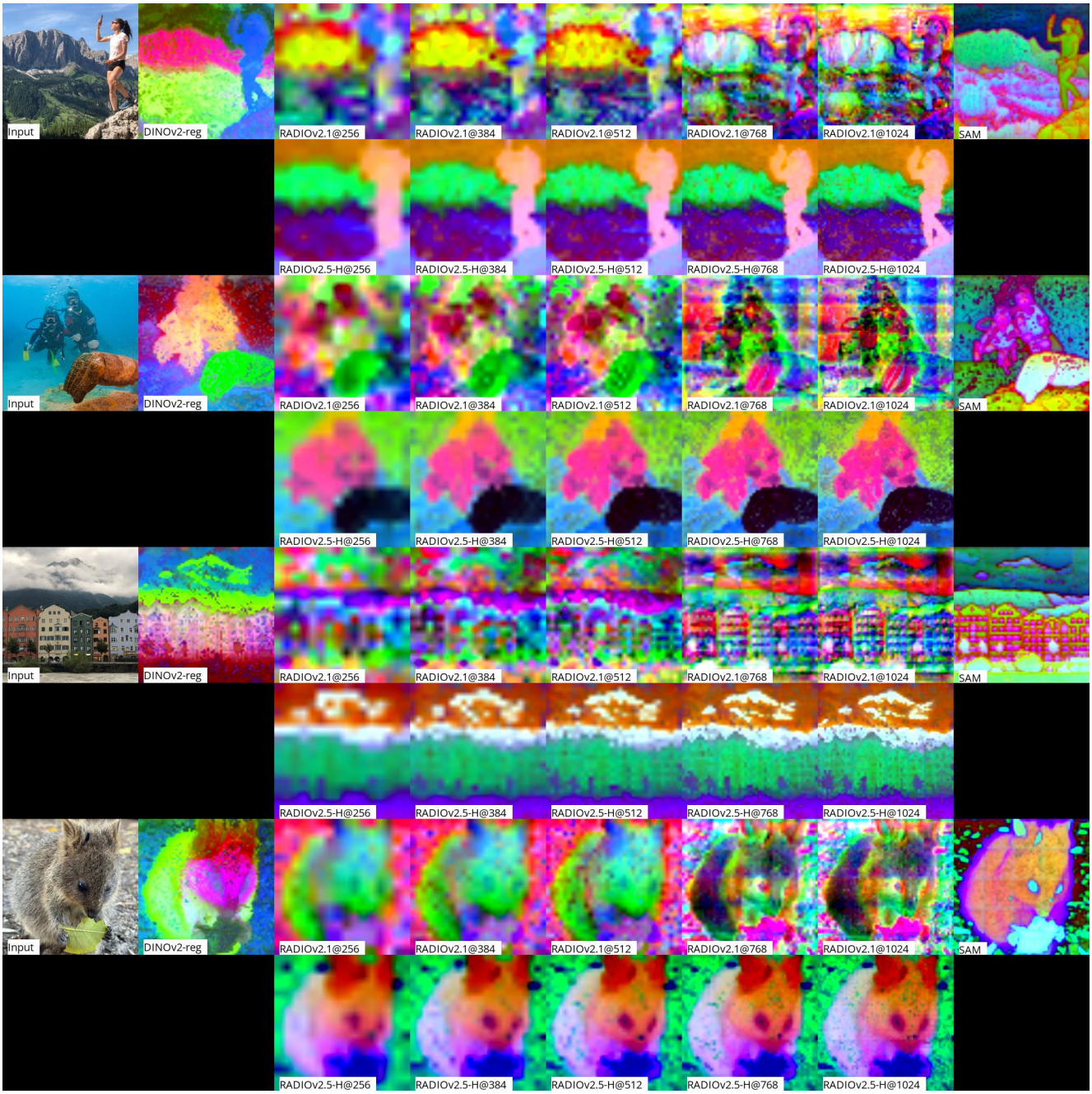

Figure 9. Visualizations of model features exhibiting the mode switch issue. We use PCA to project patch tokens into a 3D-space representing RGB colors. From left to right: input image, DINOv2, RADIO (baseline model) at 256x256, 384x384, 768x768, 1024x1024, and SAM. The visualizations illustrate how our baseline RADIO switches from producing DINO-like features at low resolution to producing SAM-like features at high resolution.

Sink Layout	SigLIP	RADIOv2.5-H	
	Values	*Keys*	*Values*
4×4	0.56	0.54	**0.50**
6×6	0.52	0.55	**0.48**
8×8	0.56	0.59	**0.53**

Table 18. Reconstruction error (normalized MSE) after token merging/unmerging, using as criteria the "keys" (we use the attention keys) or "values" (we use the patch token values). The "Sink Layout" column indicates the strided arrangements for the merge sinks. RADIOv2.5-H exhibits a systematically lower error than SigLIP.

I. Token Merging

I.1. Ablation Study on ToMe Parameters

I.2. More Token Merging Visualizations

Figure 11 shows more visualization of the ToME compression/decompression. Each input image yields 27x27=729 tokens, which are then compressed to 9 tokens using ToME.

J. Mosaic Visualizations

Figures 12 and 13 show visualizations of mosaic augmentations under 2×2 and 4×4 arrangements.

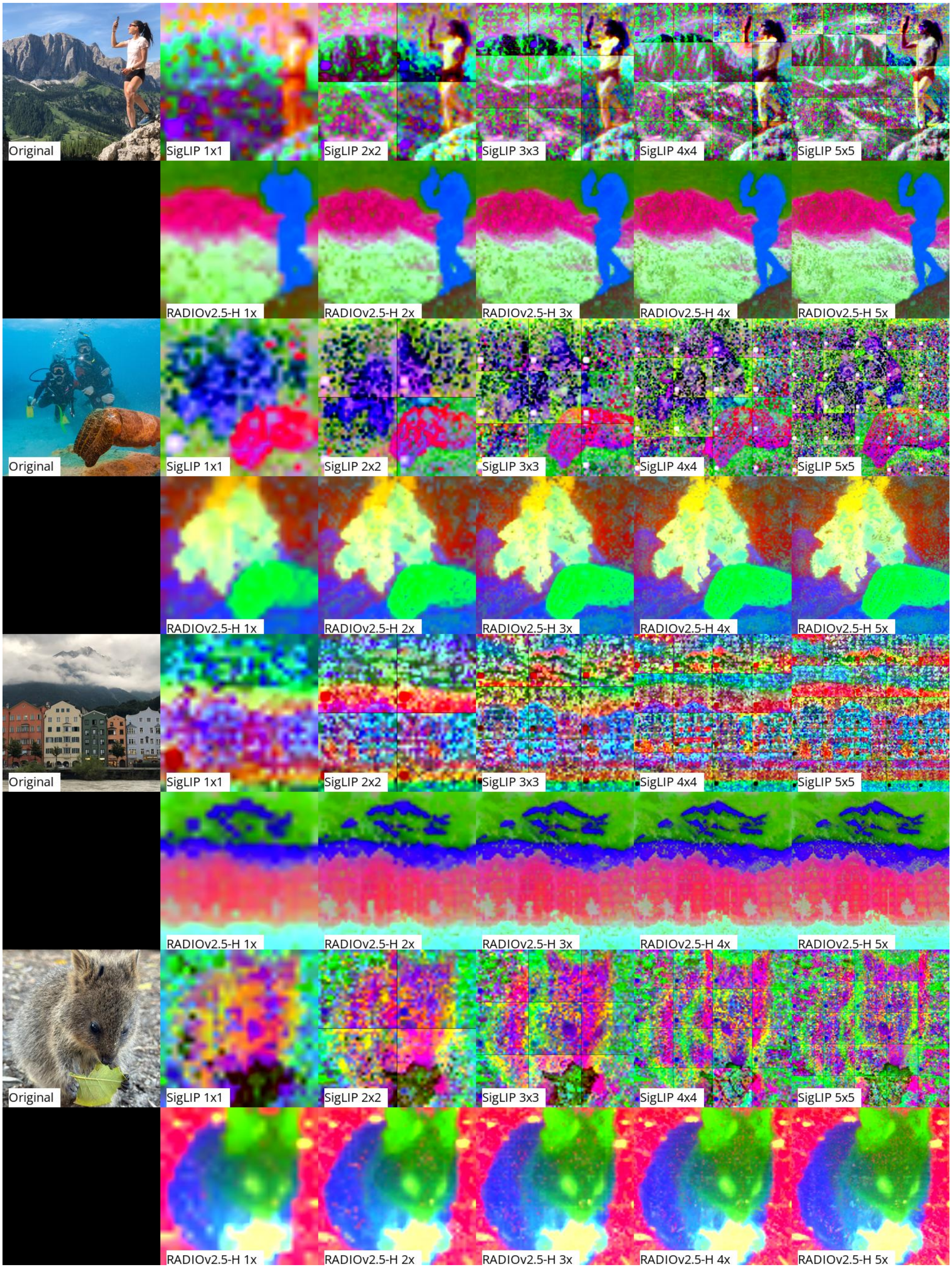

Figure 10. Visualization of image upscaling with tiling and SigLIP

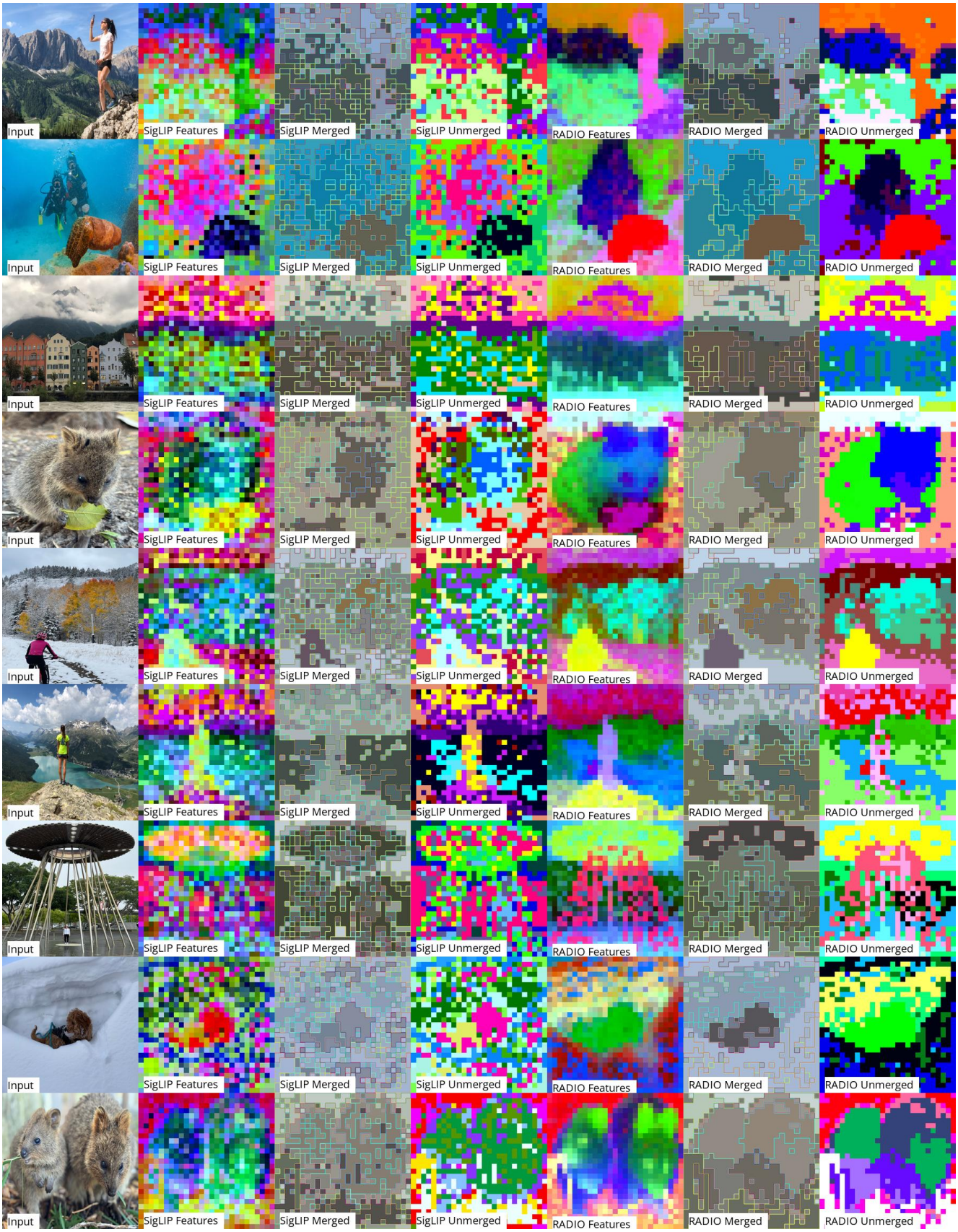

Figure 11. ToMe visualizations

Figure 12. Mosaic visualization in a 4×4 arrangement

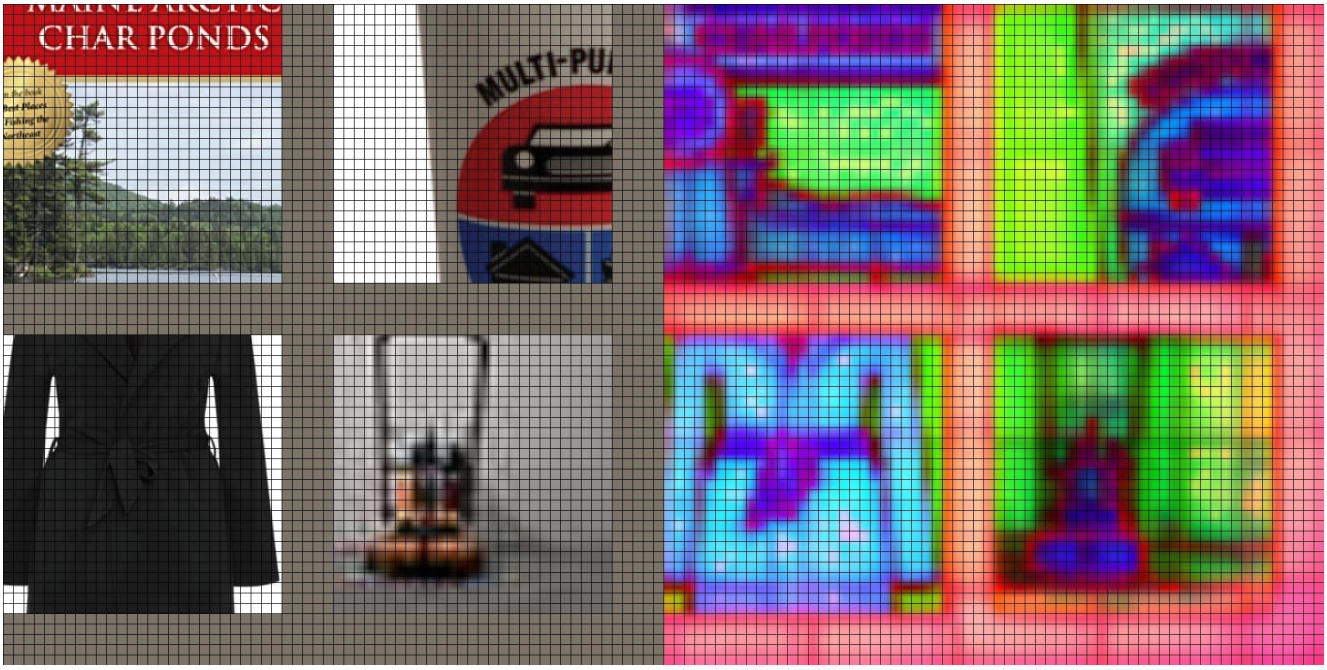

Figure 13. Mosaic visualization in a 2×2 arrangement

K. PHI Standardization (PHI-S)

K.1. Overview

PHI Standardization [34] is a method of statistical standardization that additionally prevents distortion across the channels. It is useful when you have a multidimensional distribution where all dimensions have the same units (e.g. no mixture of distance and time features). In the case of our feature matching paradigm, where all of the features we're matching are latent, this is exactly the situation we're in. In this section of the appendix, we review some of the key results of [34] in order to build an intuition of what's happening, and also why we care. In figure 14, we show the activation distributions of DFN CLIP, SAM, DINOv2, and SigLIP.

The key takeaways are:

- Every model's distributions are roughly gaussian.
- Different channels for a particular model have different centers.
- Different models have very different variances: SAM's is 29.91, and DFN CLIP's is 8.18×10^{-4}.
- If one doesn't control for these distribution differences, then the learning process biases toward the high-variance targets.
- Different normalization techniques impose tradeoffs, particularly w.r.t. distortion.
- Distortion is a problem because it forces the student to apply more weight to the low-variance dimensions of the teacher distribution, and less weight to the high-variance dimensions. Effectively the opposite of what's desired.

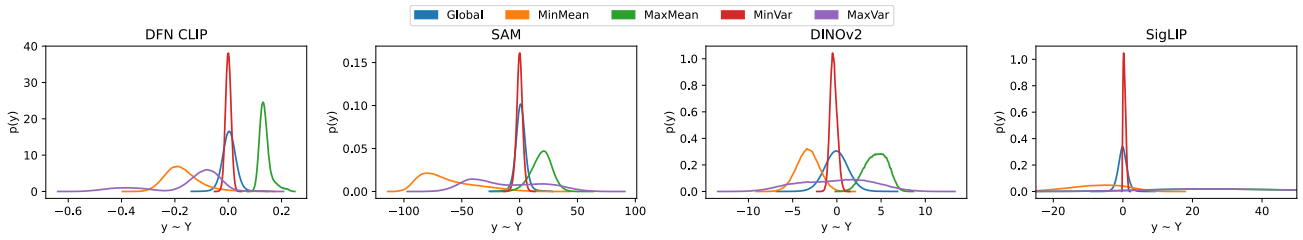

Figure 14. Teacher activation histograms. We show the global histogram, as well as the histograms for the channels associated with the minimum mean, maximum mean, minimum variance, and maximum variance. While all being roughly normal, they have very different centers and scales.

K.2. Normalization

Standardization is a standard statistical technique, where, given a multivariate distribution X_C where C is the number of dimensions, then

$$\hat{X}_i = \frac{X_i - \mu_i}{\sigma_i} \tag{12}$$

where μ_i and σ_i are the respective mean and standard deviation of dimension $i \in C$. It treats each dimension independently, which has the effect of scaling each dimension differently. The result of this method is that $\text{Var}\left[\hat{X}_i\right] = 1 \quad \forall i \in C$, and thus, at least at the start of the learning process, the expected mean-squared-error (MSE) of any channel is the same, and this is also true across the set of teachers.

PHI Standardization is the proposed technique defined as

$$\phi = \sqrt{\frac{1}{C}\sum_{i}^{C}\lambda_i} \tag{13}$$

$$\mathbf{\Lambda} = \text{diag}\left(\lambda_1, ..., \lambda_C\right) \tag{14}$$

$$\Sigma\left[\mathbf{X}\right] = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal \tag{15}$$

$$\tilde{\mathbf{X}} = \phi^{-1}\mathbf{H}\mathbf{U}^\intercal\mathbf{X} \tag{16}$$

Teacher	ϕ^2	MSE		F[X]	
		Baseline	PHI-S	Baseline	PHI-S
DFN CLIP	5.831E-4	5.100E-4	2.418E-4	1.143	**2.411**
OpenAI CLIP	0.820	0.570	0.525	1.438	**1.563**
DINOv2-g	1.729	0.222	0.206	7.799	**8.377**
SAM	27.263	3.719	5.313	**7.331**	5.132

Geometric Mean	Baseline	PHI-S
	3.114	**3.568**

Table 19. Fidelity results for each teacher, comparing between baseline and PHI-S. Higher values are better.

where $\mathbf{H} \in \mathbb{R}^{C \times C}$ is the normalized Hadamard matrix, $\Sigma[\mathbf{X}]$ is the covariance matrix of $\mathbf{X}$, and $\mathbf{U}$ and $\lambda_1, ..., \lambda_C$ are the orthogonal projection and eigenvalues of the diagonalization of $\Sigma[\mathbf{X}]$ respectively. Similar to regular standardization, $\text{Var}\left[\tilde{X}_i\right] = 1 \quad \forall i \in C$. The key difference with this technique is that $\mathbf{HU}^\intercal$ is orthogonal, and thus the only thing that affects the scale of the distribution is ϕ, which is applied identically to every dimension. This means that there is no inter-dimension distortion of the data, only a rotation and a uniform scale.

K.3. Distortion

The key point that [34] makes is that the distortion of the teacher distribution is important. It follows this construction:

If $\mathbf{W}, \boldsymbol{\mu}$ is the linear transform that is applied to the teacher distribution, $\mathbf{y}$ are the teacher features targets (pre-normalization), and $\mathbf{x}$ are the student approximation of $\mathbf{y}$, then

$$\mathbf{y} = \mathbf{W}^{-1}(\mathbf{x} + \boldsymbol{\epsilon}) + \boldsymbol{\mu} \tag{17}$$

$$= \mathbf{W}^{-1}\mathbf{x} + \mathbf{W}^{-1}\boldsymbol{\epsilon} + \boldsymbol{\mu} \tag{18}$$

where $\boldsymbol{\epsilon}$ is the approximation error. This is to say that the errors are distorted along the inverse transformation of the normalization procedure. So we get

$$\hat{\mathbf{W}} = \text{diag}\left(\frac{1}{\sigma_1}, ..., \frac{1}{\sigma_C}\right) \tag{19}$$

$$\hat{\mathbf{W}}^{-1} = \text{diag}\left(\sigma_1, ..., \sigma_C\right) \tag{20}$$

for regular standardization, and

$$\tilde{\mathbf{W}} = \phi^{-1}\mathbf{HU}^\intercal \tag{21}$$

$$\tilde{\mathbf{W}}^{-1} = \phi\mathbf{UH}^\intercal \tag{22}$$

for PHI standardization, which, in words, shows that the estimation error will be proportional to the inverse of the normalization. For regular standardization, since the normalization is per-dimension, it means that errors that occur on the channels that had the highest variance pre-standardization will be amplified, and vice versa for the small variance channels. An alternative interpretation is that the student model spends an equal amount of energy learning each channel, and thus spends too much energy optimizing low-variance dimensions at the expense of the high-variance ones, resulting in sub-optimal approximation error. In practice, $\boldsymbol{\epsilon}$ need not have uniform variance across dimensions, meaning that the student model can counteract the effect of the distortion, so in figure 15 we visualize the approximation error histograms for multiple methods (including those not mentioned in this appendix). We observe that PHI-S has the tightest control both intra and inter model compared to the other methods, owing to its lack of channel distortion. Empirically, we found that PHI-S resulted in the highest mean fidelity across the teacher set, compared to other normalization methods.

We also measure fidelity within the scope of this paper, with a quick review of the formulation:

$$F[X] = \frac{\sum_k^C \text{Var}\left[t(X)\right]}{\sum_k^C \text{Var}\left[f(X) - t(X)\right]} = \frac{\phi_i^2}{\text{MSE}(f(X), t(X))} \tag{23}$$

with $f(X)$ being the student feature distribution, and $t(X)$ being the teacher distribution. This function represents the ratio of the target distribution variance to the student's estimation error variance. A value of ≤ 1 means random sampling from the teacher distribution would be better, and ∞ would be perfect matching. We show the results of this in table 19, where it is apparent that the baseline allocates too much energy to matching SAM due it its disproportionately large distribution variance, consistent with [34]. The errors relative to the variance are overall smaller when applying PHI-S. We refer the reader to the original paper for the more thorough analysis and proofs of these normalization methods, along with others such as whitening.

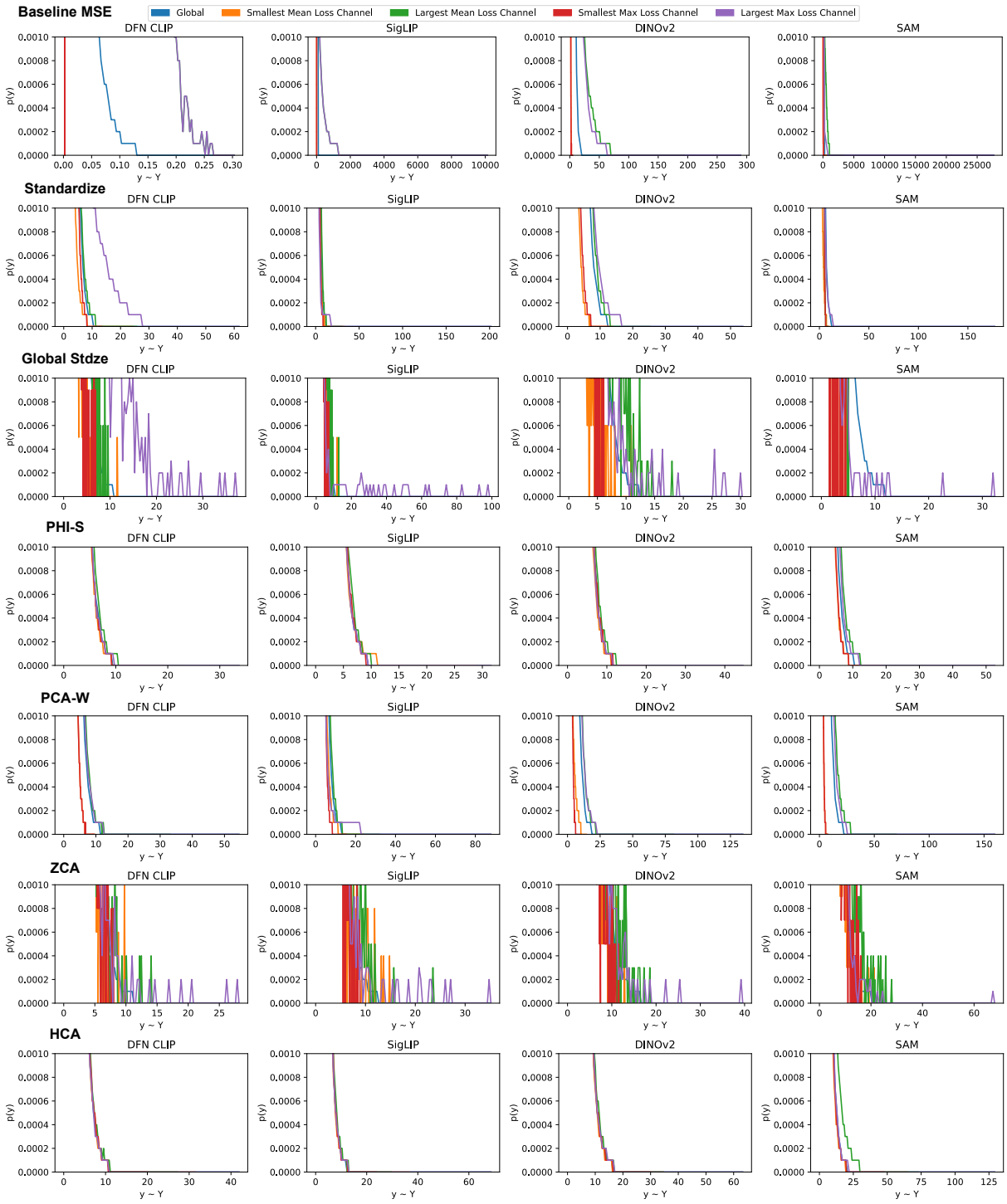

Figure 15. Loss distributions for various normalization methods. The x-axis range is based on the minimum and maximum losses seen for each method over the course of 1,000 samples after training for 100k iterations. The "Largest Max Loss Channel" shows the distribution for the channel that had the highest loss value. It helps us understand how vulnerable our learning process is to outliers. The "Global" curve shows the distribution by combining all of the channels.