Hypergraph Vision Transformers: Images are More than Nodes, More than Edges

Supplementary Material

Appendix Contents

A HgVT Model Architecture Details **2**
- A.1. Dynamic Adjacency Formation . . . 2
- A.2. Vertex Message Passing with Sparse Self-Attention . . . 3
- A.3. Hyperedge Message Passing with Fuzzy Cross-Attention . . . 3
- A.4. Sign Preserving Fuzzy Cross-Attention Modulation . . . 3
- A.5. Hypergraph Feature Processing . . . 4
- A.6. Additional Variation Options for Efficiency . . . 4

B. Computational Overhead **4**
- B.1. Improving Computational Efficiency . . . 5

C. Hypergraph Quality **5**
- C.1. Hyperedge Entropy . . . 6
- C.2. Intra-Cluster Similarity . . . 7
- C.3. Inter-Cluster Distance . . . 7
- C.4. Silhouette Score . . . 7
- C.5. Behavior with DINO Features . . . 8

D. Hypergraph Representations **9**
- D.1. Full Graph Feature Representations . . . 9
- D.2. Expert Pooling Feature Representations . . . 10

E. Graph Visualization **11**

F. Semantic Segmentation **14**
- F.1. Resolution Finetuning . . . 14
- F.2. Segmentation Results . . . 14
- F.3. Using Semantic Segmentation for Interpretability . . . 17

G Image Retrieval **17**
- G.1. Graph Pruning . . . 17
- G.2. Volumetric Similarity . . . 17
- G.3. Adaptive Reranking . . . 18
- G.4. Centroid Hashing . . . 18
- G.5. Retrieval Hyperparameter Ablations . . . 19
- G.6. Visualizing Adaptive Reranking . . . 20

H Additional Ablations **22**
- H.1. Population Regularization Sweeps . . . 22
- H.2. Correlation Analysis of Metrics . . . 24
- H.3. Expert Pooling Regularization . . . 26
- H.4. Repeated Blocks . . . 26
- H.5. Additional HgVT-Lt Model Ablations . . . 26

I. Implementation Details **28**
- I.1. Training Hyperparameters . . . 28

J. Macro-Class Clustering with Expert Edge Pooling **30**
- J.1. HgVT-Lt on ImageNet-100 . . . 30
- J.2. HgVT-S on ImageNet-1k . . . 32

A. HgVT Model Architecture Details

The Hypergraph Vision Transformer (HgVT) adapts the architecture of standard Vision Transformers by incorporating hypergraph features to enhance image analysis capabilities. Similar to Vision Transformers, HgVT utilizes a patch embedding layer as its entry point, followed by an isotropic stack of L HgVT blocks, each based on the ubiquitous Llama blocks[1] [54], culminating in feature pooling and a classifier head. Configured to process both vertex and edge information, the blocks include six main components: adjacency mask computation, vertex self-attention, edge aggregate attention, edge distribution attention, and separate feed-forward networks for vertices and edges. This configuration facilitates dynamic bipartite graph construction within each block, allowing the model to adaptively refine the input image's representative hypergraph.

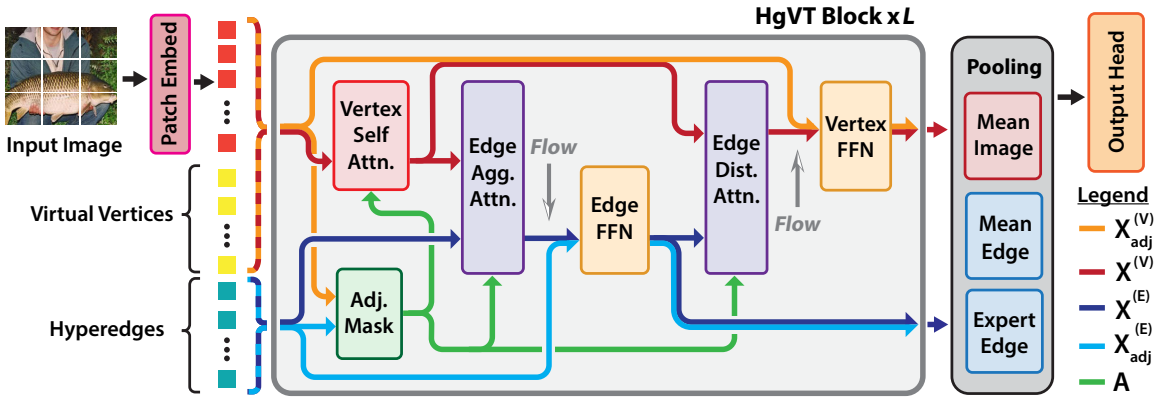


Figure 5. HgVT Architecture, composed of stacked HgVT blocks with adjacency matrix $\mathbf{A}$, vertex features $\mathbf{X}^{(V)}$, and hyperedge features $\mathbf{X}^{(E)}$. Pooling only applied to $\mathbf{X}^{(:iV)}$ and $\mathbf{X}^{(:vE)}$; edge attention flow shown with gray arrows; norms and residual omitted for clarity.

Four key feature matrices – $\mathbf{X}^{(V)}$, $\mathbf{X}^{(V)}_{\text{adj}}$, $\mathbf{X}^{(E)}$, and $\mathbf{X}^{(E)}_{\text{adj}}$ – represent the bipartite hypergraph features and are progressively updated in each HgVT block in an interleaved manner. Each block also constructs a new adjacency matrix $\mathbf{A}$ from the input $\mathbf{X}^{(V)}_{\text{adj}}$ and $\mathbf{X}^{(E)}_{\text{adj}}$ matricies, which then contributes to the attention layers within that block. As illustrated in Figure 5, this approach allows for the dynamic integration and processing of these matrices within each HgVT block, facilitating effective feature interaction and updating.

For succinct discussion in subsequent sections, the update process for each layer l is encapsulated using the following compact notation:

$$\mathbf{X}_*^{(*,l+1)} = \mathbf{X}_*^{(*,l)} + \mathbf{X}_*^{(*,l)\prime}, \quad \mathbf{X}_*^{(*,l)\prime} = f\left(\mathrm{RN}\left(\mathbf{X}_*^{(*,l)}\right), \ldots, \mathbf{A}^{(l)}\right) \tag{7}$$

where $\mathrm{RN}(\cdot)$ denotes the RMS Norm [62], and $\mathbf{X}_*^{(*,l)}$ includes both vertex features and hyperedge features, along with their respective adjacency features. The update function $f(\cdot)$ can utilize all four normalized feature matrices and the adjacency matrix $\mathbf{A}^{(l)}$, which is updated once per HgVT block.

A.1. Dynamic Adjacency Formation

Dynamically establishing the hypergraph structure is crucial for adaptability across varying semantic and spatial structures inherent in different image inputs. Mirroring the query-key interactions found in attention mechanisms, HgVT utilizes cosine similarity to evaluate the alignment between vertex and hyperedge adjacency features. This similarity assessment allows hyperedges to effectively "query" vertices for relevant features, establishing a scale-invariant comparison that focuses on the directionality of embedding vectors. To convert the cosine similarity to adjacency membership, we then form the soft adjacency matrix $\mathbf{A}$ with a sharpened sigmoid function, detailed as follows:

$$\mathbf{A} = \sigma\left(\alpha \cdot \tilde{\mathbf{X}}^{(V)}_{\text{adj}} \left[\tilde{\mathbf{X}}^{(E)}_{\text{adj}}\right]^T\right), \quad \tilde{\mathbf{X}}^{(*)}_{\text{adj}} = \frac{\mathbf{X}^{(*)}_{\text{adj}}}{||\mathbf{X}^{(*)}_{\text{adj}}||_2 + \epsilon} \tag{8}$$

where $\tilde{\mathbf{X}}^{(*)}_{\text{adj}}$ represents the L2-normalized adjacency feature matrix. Here, σ denotes the sigmoid function, and $\alpha = 4$ acts as a sharpening factor, enhancing the sigmoid's effectiveness by pushing intermediate values toward the extremes. The hard

[1] HgVT uses fixed sinusoidal position embeddings rather than rotary position embeddings.

membership adjacency matrix $\hat{\mathbf{A}} = [\mathbf{A} > 0.5]$ transforms these sigmoid outputs into binary memberships, crucial for defining significant hypergraph relationships and suitable for sparse attention masking. In configurations where feature matrices and adjacency feature matrices are tied ($\mathbf{X}^{(*)}_{\text{adj}} = \mathbf{X}^{(*)}$), $\mathbf{X}^{(*)}_{\text{adj}}$ is computed as $\mathbf{X}^{(*)}\mathbf{W}_*$, using a learned projection matrix to adapt features for adjacency computation and maintain embedding adaptability.

A.2. Vertex Message Passing with Sparse Self-Attention

Shifting from traditional hypergraph models, which typically employ a gather $\rightarrow$ scatter mechanism for processing vertex-hyperedge interactions, HgVT reconceptualizes hyperedges as communication pools that facilitate dynamic and efficient information flow among vertices and their associated hyperedges. Instead of relying on a single dense attention operation, HgVT organizes communication into two distinct streams: intra-hyperedge message passing and interactions between hyperedges and their constituent vertices. By enabling direct message passing within hyperedges, the model significantly enhances inter-vertex communication, allowing for more nuanced integration of contextual information. Furthermore, this configuration restricts interactions to vertices that share hyperedges, naturally inducing sparsity in the interaction matrix and substantially reducing computational overhead. The strategy for message passing between vertices within a hyperedge ($\mathcal{V}_e \rightarrow \mathcal{V}_e$) is implemented through the following update process:

$$\mathbf{X}^{(V)\prime} = \text{softmax}\left(\left(\mathbf{X}^{(V)}\mathbf{W}_Q\right)\left(\mathbf{X}^{(V)}\mathbf{W}_K\right)^T + \mathbf{B}\right)\left(\mathbf{X}^{(V)}\mathbf{W}_V\right), \quad \mathbf{B} = 1 - \left[\left(\hat{\mathbf{A}}\hat{\mathbf{A}}^T\right) > 0\right] \tag{9}$$

In this equation, the mask $\mathbf{B} \in \{0,1\}^{|V|\times|V|}$ is a dynamically computed based on the connectivity within the hyperedges, derived from the hard adjacency matrix $\hat{\mathbf{A}} \in \{0,1\}^{|V|\times|E|}$. This masking ensures that attention computations are confined to vertices within the same hyperedge, enhancing communication efficiency. Additionally, for simplification, the typical attention scaling factor $1/\sqrt{d_k}$, which is generally used to stabilize the softmax calculations, is omitted from the above equation.

A.3. Hyperedge Message Passing with Fuzzy Cross-Attention

Completing the concept of hyperedges as dynamic communication pools outlined in the previous section, HgVT utilizes cross-attention mechanisms to facilitate interactions between hyperedges and their constituent vertices. These mechanisms – hyperedge aggregation attention ($\mathcal{V}_e \rightarrow \mathcal{E}_e$), focusing on gathering information, and hyperedge distribution attention ($\mathcal{E}_e \rightarrow \mathcal{V}_e$), dedicated to scattering information – leverages HgVT's unique bipartite representation for effective management of information flows within these pools. By modulating the attention logits with the soft adjacency matrix $\mathbf{A}$ via a Hadamard product, the model introduces a layer of "fuzziness" to the typical cross-attention mechanism. Such modulation dynamically aligns the model's response to the varied connectivity patterns typical in hypergraph structures, thereby enhancing both precision and adaptability in processing information. The equations that formalize these attention processes are presented below:

$$\mathbf{X}^{(E)\prime} = \text{softmax}\left(\left(\mathbf{X}^{(E)}\mathbf{W}_Q\right)\left(\mathbf{X}^{(V)}\mathbf{W}_K\right)^T \circ \mathbf{A}^T + \mathbf{M}^T\right)\left(\mathbf{X}^{(V)}\mathbf{W}_V\right) \tag{10}$$

$$\mathbf{X}^{(V)\prime} = \text{softmax}\left(\left(\mathbf{X}^{(V)}\mathbf{W}_Q\right)\left(\mathbf{X}^{(E)}\mathbf{W}_K\right)^T \circ \mathbf{A} + \mathbf{M}\right)\left(\mathbf{X}^{(E)}\mathbf{W}_V\right) \tag{11}$$

In this framework, the soft adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V|\times|E|}$ modulates the attention logits through a Hadamard product ($\circ$), dynamically reflecting the true connectivity of vertices to hyperedges and providing a gradient path to update the weights used to compute the adjacency feature matrices. Concurrently, the static interaction mask $\mathbf{M} \in \{0,1\}^{|V|\times|E|}$ prevents virtual hyperedges ($v\mathcal{E}$) from interacting with image vertices ($i\mathcal{V}$), ensuring the maintenance of the hierarchical hypergraph structure described in Section 3.2 within the architecture. As before, the $1/\sqrt{d_k}$ factor is omitted for clarity.

A.4. Sign Preserving Fuzzy Cross-Attention Modulation

While simple to implement, the Hadamard modulation introduced in the previous section is sub-optimal due to properties of the softmax function, where weights of zero will bias the distribution (e.g. $e^0 = 1$). More specifically, since $A_{ij} \in [0,1)$, and we set $A_{ij} > 0.5$ to indicate membership, non-membership logits can still exhibit a positive attention contribution. Similarly, maximal dissimilarity ($A_{ij} = 0$) will move negative logits closer to zero, potentially resulting in undesirable interactions. To address this issue, we adopt sign preserving modulation, which uses the shifted adjacency form ($\tilde{\mathbf{A}} = 2\mathbf{A} - 1$), resulting in all non-membership logits becoming negative, while preserving the sign of the membership logits.

$$\tilde{\circ}(\mathbf{S}, \tilde{\mathbf{A}}) = \underset{-1\leq x\leq 1}{\text{Clamp}}\left(\text{Sign}(\mathbf{S}) + \text{Sign}(\tilde{\mathbf{A}}) + 1\right) \circ \left(\mathbf{S} \circ \tilde{\mathbf{A}}\right) \tag{12}$$

where $\mathbf{S}$ represents the pre-masked attention logits (e.g. $\mathbf{Q} \cdot \mathbf{K}^T$), and $\mathbf{A}$ is the soft adjacency matrix. The modified Hadamard product $\tilde{\circ}$ then replaces the normal Hadamard product in Eqs. (10) and (11). To better understand this functional form, we can consider the behavior table as shown in Tab. 8.

$\tilde{\mathbf{A}} \setminus \mathbf{S}$	**+**	**0**	**-**
+	$[3] \to 1 : \tilde{\mathbf{A}} \circ \mathbf{S} > 0$	$[2] \to 1 : 1 \cdot \mathbf{0} = 0$	$[1] \to 1 : \tilde{\mathbf{A}} \circ \mathbf{S} < 0$
0	$[2] \to 1 : 1 \cdot \mathbf{0} = 0$	$[1] \to 1 : 1 \cdot \mathbf{0} = 0$	$[0] \to 0 : 0 \cdot \mathbf{0} = 0$
-	$[1] \to 1 : \tilde{\mathbf{A}} \circ \mathbf{S} < 0$	$[0] \to 0 : 0 \cdot \mathbf{0} = 0$	$[-1] \to -1 : -\tilde{\mathbf{A}} \circ \mathbf{S} < 0$

Table 8. Behavior table for the modified Hadarmard product $\tilde{\circ}$. Showing how the input signs for $\tilde{\mathbf{A}} = 2\mathbf{A} - 1$ and $\mathbf{S}$ affect the output, with the pre-clamp sum in square brackets, the clamp output after the $\to$, the resultant form, and the output sign.

To implement this modified Hadamard product, we pre-compute the element-wise correction term, defined as $\phi : (a, s) \in \mathbb{R}^2 \to \{-1, 0, 1\}$, and compute the full function as a 3-element Hadamard product. Notably, this correction term has a zero derivative with respect to a and s, except at the boundaries, where it is undefined. Therefore, we can avoid complications with differentiation by applying a gradient stop to the pre-sum correction term.

A.5. Hypergraph Feature Processing

Distinct point-wise feed-forward networks (FFNs) are utilized to process vertex and hyperedge features independently within the HgVT blocks, ensuring differentiated processing for each set within the bipartite representation. These features are integrated with adjacency features through a dense, fully-connected GeGLU [47] layer, allowing the FFN to effectively combine both immediate and relational attributes. By updating both feature types and their adjacency embeddings within the same FFN layer, the model centralizes computational tasks and simplifies the message passing process by focusing solely on feature updates, avoiding the direct involvement of adjacency features and thus improving computational efficiency. The update rules are governed by the following equation:

$$\mathbf{X}_{\text{adj}}^{(*)\prime} = \text{FFN}\left(\mathbf{X}_{\text{adj}}^{(*)} || \mathbf{X}^{(*)}\right), \quad \mathbf{X}^{(*)\prime} = \text{FFN}\left(\mathbf{X}_{\text{adj}}^{(*)} || \mathbf{X}^{(*)}\right) \tag{13}$$

Here, $(*)$ represents either the vertex set $\mathcal{V}$ or hyperedge set $\mathcal{E}$, and $||$ represents concatenation.

A.6. Additional Variation Options for Efficiency

To enhance efficiency, several potential paths exist to reduce parameters and FLOPS, aligned with the principles of graph neural networks. The feature matrices $\mathbf{X}^{(*)}$ and $\mathbf{X}_{\text{adj}}^{(*)}$ can either be identical or have different dimensionalities, thereby simplifying the computational requirements of the FFN layers. Additionally, tying both FFN layers to share weights further reduces the parameter count. Consistent with practices in Graph Attention Networks [57], the weights for vertex self-attention (W_Q, W_K, W_V) and edge cross-attention (W_K, W_V) can also be tied. Implementing these strategies offers a range of options to tailor HgVT variants for balancing memory usage and computational efficiency, optimizing the model for various deployment environments based on performance needs.

B. Computational Overhead

In this section, we explore the computational overhead of our proposed HgVT models relative to other isotropic models. All benchmarking experiments were conducted on an NVIDIA Quadro RTX 4000 GPU, using PyTorch 2.5.1 with CUDA 12.2. We evaluated all models in 32-bit precision with a batch size of 32. To ensure stable measurements, we aggregated statistics over 100 iterations, following an initial 10 warmup iterations to mitigate the impact of GPU initialization overhead. The comparative results are presented in Tab. 9, which also includes a detailed cost breakdown, summed over all layers of the same type. For completeness, we also report computational performance for a theoretical HgVT-B model ($d_f = 448$, $d_a = 128$, $L = 16$, $h = 14$) that was not trained but included to illustrate its expected cost. Notably, we were unable to benchmark ViHGNN [17] due to reproducibility issues with the publicly released code and have therefore excluded it from the table.

The results are summarized in Tab. 9, where models are grouped by scale and ordered by Top-1 ImageNet accuracy. The main points of comparison are other vision transformers [2, 12, 53] and ViG [16], a graph convolution-based model. We find that both ViG and HgVT exhibit higher latency and lower throughput than comparable vision transformers. For ViG, this increased cost is attributed to the computationally expensive graph convolution operations. In the case of HgVT, the increased

Table 9. Comparison of inference performance for HgVT and other isotropic networks. All results measured using 32-bit precision and a batch size of 32 on an NVIDIA Quadro RTX 4000 GPU. Further showing time per component and overall inference percentage, with Spatial denoting either Self-Attention or Graph Conv layers. ♦Transformer, ★GNN, and ▲HgVT. †Hypothetical HgVT-B model (not trained).

Model	Params	FLOPs	ImageNet Top-1	ImageNet Top-5	VRAM (MB) Static	VRAM (MB) Peak	Batch Time (ms)	Speed (imgs/s)	Time per Component (ms) Patch	Spatial	FFN	Cluster	Aggregate	Distribute
♦DeiT-Ti [53]	5.7M	1.3B	72.2	91.1	48.5	104	23.2 ±0.1	1370 ±8	0.6 (2.6%)	9.7 (41%)	9.5 (40%)	–	–	–
★ViG-Ti [16]	7.1M	1.3B	73.9	92.0	54.3	331	79.5 ±0.3	402 ±1.6	3.0 (3.8%)	49.4 (62%)	13.5 (17%)	12 (15%)	–	–
▲HgVT-Mi (ours)	5.8M	1.4B	74.4	92.2	49.2	203	36.4 ±0.2	880 ±4.5	2.5 (7.0%)	6.8 (19%)	12.1 (33%)	1.1 (3.0%)	3.4 (9.4%)	3.8 (10%)
▲HgVT-Ti (ours)	7.7M	1.8B	76.2	93.2	56.6	210	47.1 ±0.2	679 ±3.0	2.5 (5.4%)	9.0 (19%)	16.0 (34%)	1.5 (3.1%)	4.5 (9.6%)	5.0 (10%)
♦DINOv1-S [2]	21.7M	4.6B	77.0	–	119	249	68.4 ±0.4	468 ±2.5	1.2 (1.7%)	29.6 (43%)	32.8 (48%)	–	–	–
♦DeiT-S [53]	22.1M	4.6B	79.8	95.0	111	223	64.3 ±0.4	498 ±2.7	1.1 (1.8%)	25.7 (40%)	32.3 (50%)	–	–	–
★ViG-S [16]	22.7M	4.5B	80.4	95.2	114	573	191 ±1.1	168 ±0.9	6.5 (3.4%)	120 (63%)	41.7 (22%)	20 (11%)	–	–
▲HgVT-S (ours)	22.9M	5.5B	81.2	95.5	116	365	113 ±0.5	282 ±1.3	6.0 (5.3%)	22.4 (20%)	45.9 (41%)	1.9 (1.7%)	11 (10%)	11 (9.9%)
♦ViT-B/16 [12]	86.4M	55.5B	77.9	–	372	633	221 ±1.3	145 ±0.9	2.3 (1.0%)	87.6 (39%)	126 (56%)	–	–	–
♦DeiT-B [53]	86.4M	17.6B	81.8	95.7	357	579	213 ±1.3	150 ±0.9	2.2 (1.0%)	80.2 (37%)	124 (58%)	–	–	–
★ViG-B [16]	86.8M	17.7B	82.3	95.9	359	1271	449 ±4.7	71.2 ±0.7	20 (4.4%)	281 (61%)	127 (28%)	27 (5.8%)	–	–
▲HgVT-B (ours)†	87.9M	20.4B	–	–	367	813	323 ±3.0	99.0 ±0.9	18 (5.6%)	56.0 (17%)	157 (49%)	2.5 (0.8%)	31 (9.5%)	32 (9.6%)

cost stems from the second FFN layer (used for edges) and the additional attention operations for aggregation and distribution steps (as part of the bipartite hypergraph communication pool framework). However, despite this added complexity, HgVT remains within $2\times$ the performance of vision transformers. Notably, HgVT demonstrates lower self-attention cost despite operating on a larger sequence length (246 vs 196 for a 224^2 resolution), resulting from the reduced hidden dimension.

We also find that the expert edge pooling strategy has a negligible effect on inference performance (accounting for less than 0.3% of total inference cost), and HgVT's regularization strategy exhibits no inference cost, as it is only used to learn how to construct well-structured hypergraphs that can generalize at inference time. When comparing with ViG, HgVT consistently outperforms in both throughput ($1.4 \times -1.6\times$) and peak memory usage ($0.6\times$). Finally, HgVT's implicit clustering approach is an order of magnitude faster than ViG's KNN-based clustering, highlighting the benefits of learned self-sparsification with dynamic regularization over explicit clustering.

B.1. Improving Computational Efficiency

Although hypergraph-based models are often perceived as computationally expensive, HgVT demonstrates competitive performance and memory efficiency, outperforming ViG in both throughput and peak memory usage. However, further improvements in computational efficiency are possible through targeted optimizations. One promising direction is reordering the hypergraph block structure to enable more efficient batched matrix multiplications. This could potentially reduce the cost of the second FFN layer by up to 50%. A significant source of overhead stems from the split representations ($\mathbf{X}^{(V)}, \mathbf{X}^{(E)}, \mathbf{X}^{(V)}_{\text{adj}}, \mathbf{X}^{(E)}_{\text{adj}}$), which require multiple normalization and matrix multiplication steps that could be combined or parallelized. Additionally, the sparsity properties of the attention mechanism – including diagonal symmetry in self-attention and sparsity in edge attention – could be further leveraged through custom kernels. Moreover, the benefits from sparsity are expected to scale more effectively at higher resolutions, where the cost of attention operations grows quadratically with sequence length.

Alternatively, larger models may reduce the performance gap, as illustrated by the hypothetical HgVT-B model shown in Tab. 9. The smaller gap in inference performance for HgVT-B suggests that increasing the computational workload per operation helps mitigate the relative impact of the call-graph overhead from the split representations. This indicates that scaling the model size may naturally improve computational efficiency by better amortizing fixed costs.

C. Hypergraph Quality

To understand the structural quality of the generated hypergraph in HgVT, we consider how effectively it organizes features into coherent, distinct clusters. Unlike fully connected transformer architectures, HgVT uses hypergraphs to structure relationships in a way that preserves sparsity while capturing feature groupings. However, a naïve approach may achieve high-quality metrics on trivial tasks, strongly aligning with low-level features (such as textures) rather than assessing the model's ability to capture more nuanced structural qualities. We therefore propose using four key metrics – Hyperedge Entropy, Intra-Cluster Similarity, Inter-Cluster Distance, and Silhouette Score – to achieve a balanced assessment, while ensuring that these metrics are computationally feasible and well-defined for practical evaluation.

In the context of HgVT, a "cluster" corresponds to a primary hyperedge ($p\mathcal{E}$) within the hypergraph, where virtual hyperedges ($v\mathcal{E}$) are excluded due to the hierarchical graph structure. Each primary hyperedge represents a grouping of vertices $\mathcal{V} = i\mathcal{V} \cup v\mathcal{V}$, where we primarily focus on image vertices ($i\mathcal{V}$). This approach excludes virtual vertices ($v\mathcal{V}$), which serve as summarization tokens and are expected to be largely distinct from the image vertices due to the diversity regularization.

By defining clusters through primary hyperedges, we focus our evaluation on image-based feature groupings, assessing the quality of these groupings with respect to the specific properties captured by the following metrics.

1. **Hyperedge Entropy (HE):** Assesses the internal diversity within clusters.
2. **Intra-Cluster Similarity (ICS):** Measures cohesion among vertices within clusters.
3. **Inter-Cluster Distance (ICD):** Evaluates separation between clusters.
4. **Silhouette Score (SIL):** Provides an overall measure of clustering quality, balancing cohesion and separation.

The groupings within each primary hyperedge ($p\mathcal{E}$) are defined with fuzzy weights derived from the soft adjacency matrix $\mathbf{A}$, which encodes the membership strength between vertices and hyperedges. All clustering quality metrics are therefore exclusively computed on the vertex features $\mathbf{X}^{(V)}$, where using the image subset $\mathbf{X}^{(:iV)}$ allows for direct correspondence with strong vision embeddings, such as from DINOv2 [38] and CLIP [42]. Furthermore, many of the metrics can be simplified to utilize cluster centroids, resulting in more computationally efficient computations. For a given cluster j, the centroid $E_{c,j}$ is calculated as:

$$E_{c,j} = \frac{\sum_{k \in \mathcal{V}} \mathbf{A}_{kj} X_k}{\sum_{k \in \mathcal{V}} \mathbf{A}_{kj}} \tag{14}$$

where $X_k \in \mathbb{R}^d$ represents the feature vector for the k-th vertex in $\mathcal{V}$. This centroid formulation leverages the soft adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V| \times |E|}$ to weigh each vertex's contribution proportionally to its membership strength to the j-th hyperedge.

To standardize notation, we define two common functions for cosine similarity (csim) and distance (cdist), which are used throughout the metrics. Cosine similarity between two feature vectors X_i and X_j is defined as:

$$\mathrm{csim}(X_i, X_j) = \frac{X_i \cdot X_j}{||X_i||\ ||X_j||} \tag{15}$$

where $\cdot$ represents the dot product, and $||X||$ denotes the L2 norm of X. Likewise, cosine distance – used as a measure of dissimilarity – is defined:

$$\mathrm{cdist}(X_i, X_j) = 1 - \frac{X_i \cdot X_j}{||X_i||\ ||X_j||} \tag{16}$$

C.1. Hyperedge Entropy

Hyperedge Entropy (HE) measures the concentration of vertex features within each cluster (hyperedge), quantifying how "focused" or homogeneous the feature distribution is within each cluster. Using entropy provides a measure of intra-cluster coherence, capturing the spread of vertex feature similarities with respect to the centroid feature for each hyperedge.

To compute HE for a given hyperedge j, we first calculate the cosine similarity between each vertex feature X_i and the centroid feature $E_{c,j}$ of the cluster. This similarity score quantifies the alignment between individual vertex features and the core representation of the cluster. We then define p_{ij} as a normalized similarity score, computed using a softmax function over these cosine similarities, limited to vertices belonging to the cluster j as defined by the hard adjacency matrix $\hat{\mathbf{A}}$:

$$p_{ij} = \frac{\exp(\mathrm{csim}(X_i, E_{c,j}))}{\sum_{v \in \mathcal{E}_j} \exp(\mathrm{csim}(X_v, E_{c,j}))} \tag{17}$$

where $\mathcal{E}_j$ represents the set of vertices (indexed by v) in the j-th hyperedge as defined by $\hat{\mathbf{A}}$. The entropy for each hyperedge j is then calculate as:

$$\mathrm{HE}_j = -\sum_{i \in \mathcal{E}_j} p_{ij}\ \log(p_{ij}) \tag{18}$$

This formulation yields an entropy distribution over the $|\mathcal{E}|$ hyperedges for a given graph, and a larger distribution when aggregated over an evaluation dataset. Here, lower entropy values indicate more concentrated, homogeneous feature distributions within the cluster, and higher entropy suggests more diverse or spread-out feature distributions.

From an interpretive standpoint, low HE values may signal that the cluster is dominated by homogeneous features, often associated with low-level structures, such as texture. For instance, in an image of a cat, a hyperedge with a low HE could indicate that fur-related features are overly concentrated, which may reflect a focus on surface-level details rather than high-level semantic structure. Conversely, a high HE can indicate poor intra-cluster coherence or semantic clustering, potentially caused by noise or irrelevant feature vectors within the cluster. Thus, balancing HE across clusters is desirable to ensure that hyperedges reflect meaningful, well-structured groupings of image features.

C.2. Intra-Cluster Similarity

Intra-Cluster Similarity (ICS) measures the cohesion of vertex features within each cluster (hyperedge), providing a sense of how similar the features are within each group. For each hyperedge, ICS is calculated as the average cosine similarity between each vertex feature X_i and the centroid feature $E_{c,j}$ of the j-th hyperedge. This metric captures the internal consistency of each cluster, with higher values indicating more cohesive feature groupings.

$$\mathrm{ICS}_j = \frac{1}{|\mathcal{E}_j|} \sum_{i \in \mathcal{E}_j} \mathrm{csim}(X_i, E_{c,j}) \tag{19}$$

where $\mathcal{E}_j$ represents the set of vertices (index by i) in hyperedge j as defined by the hard adjacency matrix $\hat{\mathbf{A}}$. To ensure meaningful results, clusters with fewer than two vertices are omitted from this calculation, as they lack sufficient members to define intra-cluster similarity.

C.3. Inter-Cluster Distance

Similar to ICS, Inter-Cluster Distance (ICD) measures how distinct different clusters (hyperedges) are from one another. Specifically, ICD quantifies the separation between clusters by measuring the cosine distance between the centroids of hyperedge pairs. This metric reflects how far apart different clusters are in feature space, with higher values indicating greater separation and, thus, more distinct feature groupings. For each pair of hyperedges (j, k), ICD is computed as:

$$\mathrm{ICD}_{j,k} = \mathrm{cdist}(E_{c,j}, E_{c,k}) \tag{20}$$

The overall ICD for the graph can then be aggregated by taking the average distance across all hyperedge pairs:

$$\mathrm{ICD} = \frac{1}{|\mathcal{E}|(|\mathcal{E}| - 1)} \sum_{j \neq k} \mathrm{ICD}_{j,k} \tag{21}$$

C.4. Silhouette Score

The Silhouette Score [45] combines both intra-cluster similarity (cohesion) and inter-cluster distance (separation) to provide an overall measure of the clustering quality. This score evaluates how well each vertex is clustered with respect to its assigned hyperedge and nearby clusters. For each vertex i within a hyperedge j, two values are defined:

- a_{ij}: the average distance between vertex i and all other vertices within its assigned hyperedge j, computed using the soft adjacency matrix $\mathbf{A}$.
- b_{ij}: the lowest average cosine distance between vertex i and all vertices in other hyperedges, effectively measuring how close i is to its nearest neighboring cluster.

These two values are calcualted as follows:

$$a_{ij} = \frac{\sum_{v \in \mathcal{E}_j, v \neq i} \mathbf{A}_{vj} \, \mathrm{cdist}(X_i, X_v)}{\sum_{v \in \mathcal{E}_j, v \neq i} \mathbf{A}_{vj}} \tag{22}$$

$$b_{ij} = \min_{k, k \neq j} \frac{\sum_{v \in \mathcal{E}_k, v \neq i} \mathbf{A}_{vj} \, \mathrm{cdist}(X_i, X_v)}{\sum_{v \in \mathcal{E}_k, v \neq i} \mathbf{A}_{vj}} \tag{23}$$

The Silhouette Score s_{ij} for the i-th vertex in the j-th hyperedge is computed as:

$$s_{ij} = \frac{b_{ij} - a_{ij}}{\max(a_{ij}, b_{ij})} \tag{24}$$

where the individual score s_{ij} is bounded by $[-1, 1]$, where more positive indicates strong cluster cohesion, more negative indicates poor clustering, and zero indicates that the vertex lies on the boundary between clusters. Finally, the global Silhouette score for the graph can be computed by averaging s_{ij} across all edges and vertices:

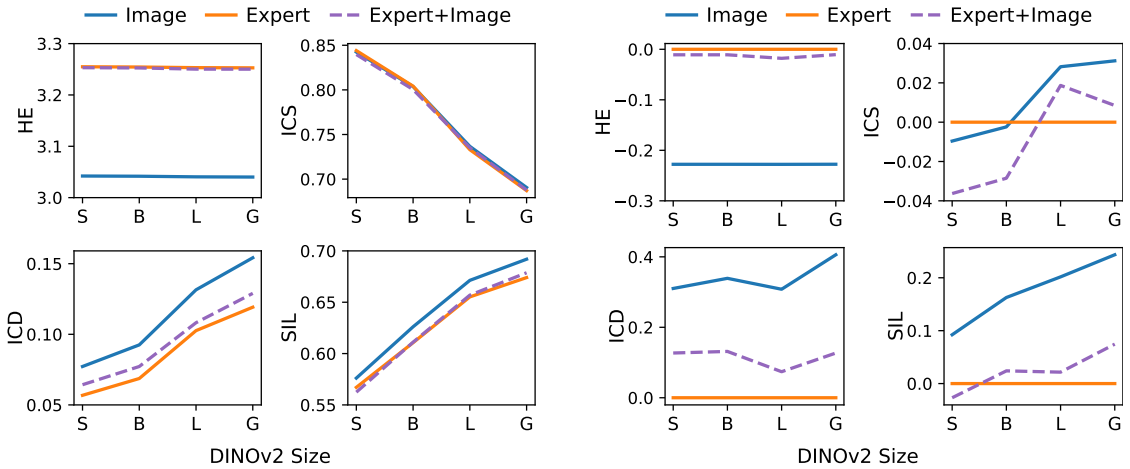
$$\mathrm{SIL} = \frac{1}{|\mathcal{E}| \, |\mathcal{V}|} \sum_{j \in \mathcal{E}} \sum_{i \in \mathcal{V}} s_{ij} \tag{25}$$

The global Silhouette score omits clusters with fewer than two elements (as is standard), due to s_{ij} being undefined for such pairs. From an interpretive standpoint, a higher SIL (closer to one) is ideal; however, it too can suffer from the same flaw as HE and ICS, where focus on trivial (texture) clustering result in better values, incorrectly suggesting strong clustering. Similarly, highly sparse graphs with small vertex counts per hyperedge can result in higher than expected SIL scores. For example, it is easier to form a tight cluster of two vertices than 20. We therefore suggest considering all four metrics en-aggregate, where a high SIL score is only meaningful with a high ICS, ICD, and a moderate to high HE (indicating diversity within each cluster).

C.5. Behavior with DINO Features

Following the definitions of graph quality metrics, we explore how these metrics behave when HgVT-Lt's feature representations are substituted with DINOv2 [38] features of progressively richer semantic strength. This analysis serves two purposes. First, it allows us to validate our chosen graph quality metrics by observing whether they effectively capture structural differences as feature richness increases, supporting the interpretive value of these metrics within the HgVT framework. Second, it provides insight into the level of semantic detail the HgVT model's hypergraphs are focusing on, shedding light on the model's capacity to capture and represent varying levels of semantic information.

We consider three pooling methods – image pooling, expert pooling, and combined pooling – within the HgVT-Lt model trained on ImageNet-100. Image pooling considers only image vertices ($i\mathcal{V}$), ignoring the hypergraph structure; expert pooling incorporates hierarchical information flow through virtual hyperedges ($v\mathcal{E}$); and combined pooling integrates both approaches. For each configuration, we extract the hypergraphs of all ImageNet-100 validation images (totaling 5k). Specifically, we utilize the soft adjacency matrix $\mathbf{A}$ from the final layer of HgVT-Lt and then substitute the image vertex features $\mathbf{X}^{(:iV)}$ with the final DINOv2 features (spanning model scales S, B, L, G). Notably, the the pooling methods indirectly influence the hypergraph structure, as they primarily affect the classification head during training but subsequently affect the generated hypergraph structure through learned representations.



(a) Metric Scaling.

(b) Cliff's Delta Metric Scaling.

Figure 6. Comparing graph quality metrics under DINOv2 feature scaling with HgVT-Lt trained on ImageNet-100. Further comparing expert, image, and combined pooling methods. Showing (a) the raw metric medians, and (b) the Cliff's D measure for the metric distributions against expert pooling as a basline.

The spatial correspondence of both DINOv2 features and HgVT's image vertices with the original input image allows us to substitute the original HgVT image vertex features with DINOv2 features. This alignment is well-established in applications such as object segmentation and depth estimation for DINOv2 features and verified through graph visualizations in Appendix E for HgVT. To preserve spatial coherence between the two models, we resize DINO input images to 280x280 from the original 160x160 resolution. With a patch size of 14, this resizing yields 20x20 image tokens, which are then aggregated using 2x2 patches to match HgVT-Lt's 10x10 image vertex structure. For each pooling configuration, we compute the graph quality metrics across DINOv2 model scales (as shown in Fig. 6a) and measure the effect sizes of these distributions, using expert pooling as a baseline with Cliff's Delta for comparison (see Fig. 6b). Cliff's Delta [5] provides a non-parametric measure of effect size that quantifies the degree of separation between two distributions, with values close to 0 indicating minimal difference and values approaching ± 1 indicating strong differences in distribution. Notably, all measured distributions exhibit

a statistically significant separation, as measured by a K-S test.

As DINO model size increases, all pooling methods exhibit consistent trends in clustering metrics. Hyperedge Entropy (HE) remains stable, indicating that the overall spread of feature diversity within clusters is unaffected by feature scaling. However, Intra-Cluster Similarity (ICS) decreases, revealing finer distinctions within existing clusters as DINO features scale. Meanwhile, Inter-Cluster Distance (ICD) and Silhouette Score (SIL) increase, reflecting improved separation among the fixed clusters. These trends suggest that as DINO models grow, they approach a more balanced clustering structure, similar to HgVT's (with ICS around 0.45 and ICD around 0.32). This convergence implies that HgVT may capture a level of semantic structure comparable to what would be achieved by a much larger DINO model, highlighting HgVT's inherent efficiency in representing semantically rich information.

When comparing pooling methods, differences in clustering metrics for expert and image pooling remain mostly consistent (when considering effect size). Image pooling yields marginally higher ICS with increasing DINO model size and noticeably higher ICD and SIL, resulting in clusters that are more cohesive and well-separated. This suggests that image pooling may focus on distinct, cohesive textures, with reduced graph inter-connectivity and cluster overlap. Expert pooling, by contrast, exhibits higher HE and lower ICD, indicating that clusters are more internally diverse and less distinctly separated. In this case, omitting the image vertices during classification allows for increased graph connectivity, which is reflected by a degradation of clustering metrics. Finally, the combined pooling method aligns closely with expert pooling, while recovering a slight improvement to ICD and SIL due to the direct inclusion of image vertices during classification.

D. Hypergraph Representations

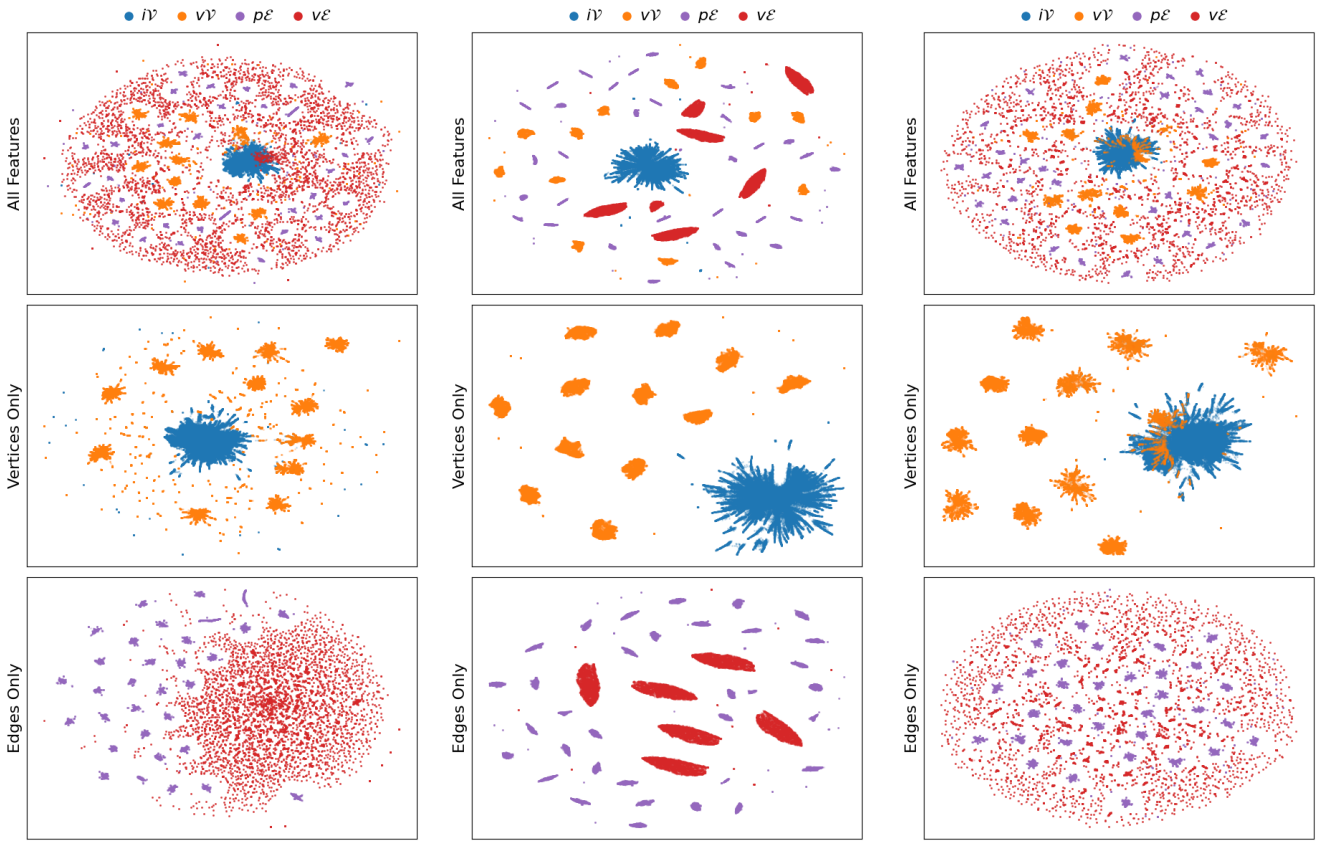
In this section, we explore the spatial organization of feature representations using Uniform Manifold Approximation and Projection (UMAP) visualizations [34], generated from the HgVT-Lt model on the ImageNet-100 validation set. UMAP enables a comparative analysis of how different components within the hypergraph structure distribute features in their learned latent space. By reducing dimensionality to two components, UMAP highlights the spatial clustering of graph feature vectors, extracted from the model's final layer. To address varying group sizes ($i\mathcal{V}$, $v\mathcal{V}$, $p\mathcal{E}$, $v\mathcal{E}$), we standardize each plot's sample size to the minimum group size, randomly sampling from other groups as needed to ensure consistency.

D.1. Full Graph Feature Representations

We explore the full graph feature representations by considering all features ($\mathcal{V} \cup \mathcal{E}$), only vertices ($\mathcal{V}$), and only edges ($\mathcal{E}$) across three pooling methods: expert pooling, image pooling, and a combined approach. The UMAP representations shown in Fig. 7 utilize a nearest neighbors setting of 10 and a minimum distance of 0.1, with consistent seeds for reproducibility.

From the UMAP results, we observe a distinct separation between expert and image pooling, with the combined method exhibiting characteristics of both. In all cases, image vertices ($i\mathcal{E}$) form relatively tight clusters, typically surrounded by other feature categories. Distinct clusters are evident for virtual vertices ($v\mathcal{V}$) and primary hyperedges ($p\mathcal{E}$), with 12 ($|v\mathcal{V}|$) and 32 ($|p\mathcal{E}|$) clusters, respectively. Under image pooling, virtual hyperedges ($v\mathcal{E}$) form six ($|v\mathcal{E}|$) distinct groups, likely due to the absence of model incentives to leverage these features for classification. In contrast, in the expert and combined pooling cases, virtual hyperedges appear as diffuse clouds, suggesting strong interconnectivity with virtual vertices and primary hyperedges.

For expert pooling, virtual hyperedges show overlap with image vertices, a phenomenon absent in the combined pooling case. This overlap likely represents low-level image features that must be transmitted through virtual hyperedges in the expert pooling scenario, whereas in the combined case, they can be transmitted directly through pooled image features. Additionally, we observe diffuse overlap of virtual vertices with image vertices in expert pooling, replaced by a single overlapping virtual vertex in the combined case. This distinction suggests two possible strategies for supporting lower-level image features: either a shared overlap across virtual vertices or a single dedicated virtual vertex providing feature support. Overall, the UMAP results align with the findings from the previous section.



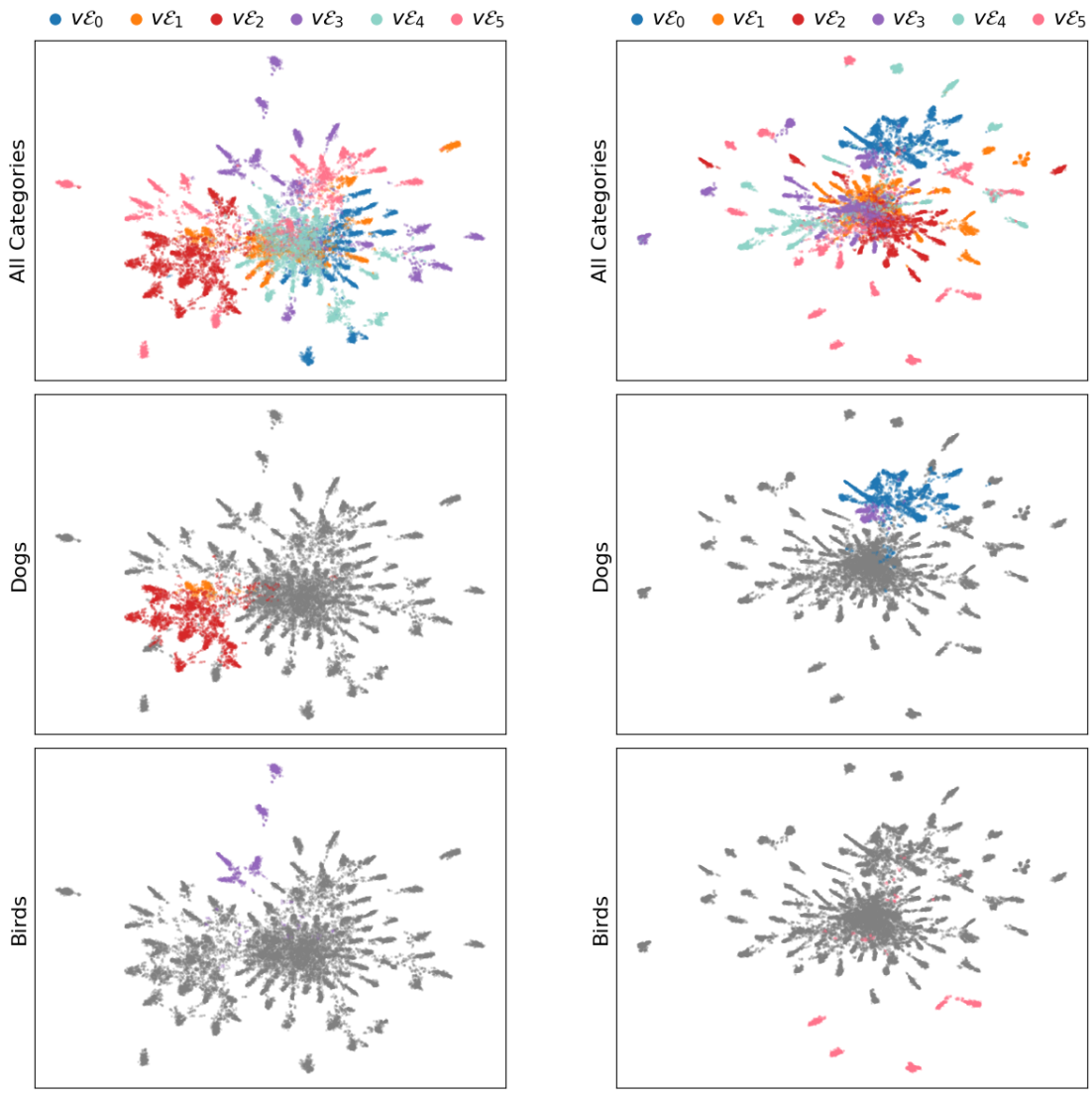
(a) Expert Pooling. (b) Image Pooling. (c) Expert + Image Pooling.

Figure 7. UMAP plots of the HgVT-Lt model under different pooling methodings: (a) Expert pooling, (b) Image pooling, and (c) both Expert and Image pooling. Showing image vertices ($i\mathcal{V}$), 12 virtual vertices ($v\mathcal{V}$), 32 primary hypereges ($p\mathcal{E}$), and 6 virtual hyperedges ($v\mathcal{E}$).

D.2. Expert Pooling Feature Representations

Given the clustering behavior for virtual edges ($v\mathcal{E}$) observed in the previous section, we further examine their structure when plotted independently to determine if unique patterns emerge. Specifically, we assess whether this structure correlates with specific experts (edge IDs) or macro-classess, such as Dogs and Birds in ImageNet-100, considering both the expert pooling and combined cases. Due to the diffuse nature of this feature type, we increase the nearest neighbors setting to 120 and set the minimum distance to 0.5 for clearer clustering in Fig. 8.

The clustering of edge IDs suggests that specific edges capture both overlapping and distinct aspects of the feature space, with each cluster representing shared or distinct features specialized for certain macro-classes. This behavior is validated when considering the clusters corresponding to the dog macro-class, emerging in both the expert and combined pooling cases. In contrast, when considering birds, they consistently form a less compact cluster, occupying a unique sub-region with minimal interference from other categories. Notably, bird features are more tightly clustered in the expert pooling case, while in the combined pooling case, bird features are more dispersed, with some overlapping with the center. This increased spread in the combined case likely reflects the distributed influence of expert edges, which only partially contribute to the final clusters, whereas the expert-only case preserves more focused class-specific features. Additionally, we observe that birds consistently align with a single expert ID, while dogs are associated with no more than two expert IDs. This allocation pattern is further analyzed in Appendix J.



(a) Expert Pooling.

(b) Expert + Image Pooling.

Figure 8. UMAP plots of virtual hyperedge classification allocation for the HgVT-Lt model under different pooling methodings: (a) Expert pooling, (b) both Expert and Image pooling. Showing overall expert allocation $v\mathcal{E}_i$, and select ImageNet-100 macro-classes: Dogs and Birds.

E. Graph Visualization

Visualizing the hypergraph structure in HgVT provides crucial insights into how various components – such as virtual vertices, primary hyperedges, and image vertices – interact to inform predictions. However, given the complexity of hypergraphs and the dense interconnections across vertices (nodes) and edges, a straightforward visualization would be overwhelming and challenging to interpret. To address this, we apply a pruned projection method that represents the hypergraph in "slices," focusing on key relationships while filtering out less influential components. This approach balances interpretability with structural fidelity, offering a clearer view of the hypergraph's hierarchical organization.

In this method, we begin by selecting the top-1 (most confident) virtual edge as the root node. From this root, we identify and rank the connected virtual vertices (vNodes) using the soft adjacency matrix $\mathbf{A}$, selecting those with contributions above a threshold of 0.1. For each vNode, we identify the top-H primary hyperedges (pEdges) and treat each as an individual slice in the visualization. Some pEdges appear in the top-H of multiple vNodes, enabling the visualization to capture overlapping and shared feature pathways effectively. Each pEdge is visualized as a 2D image, with patch dimming based on contribution

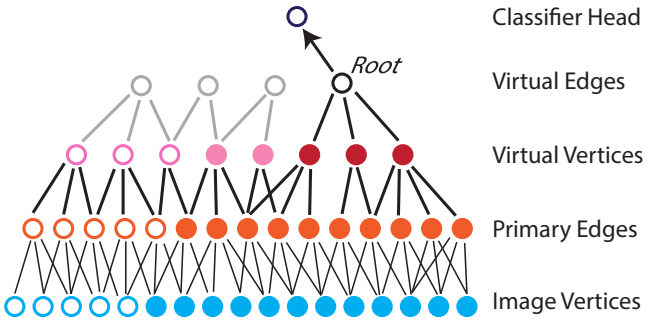
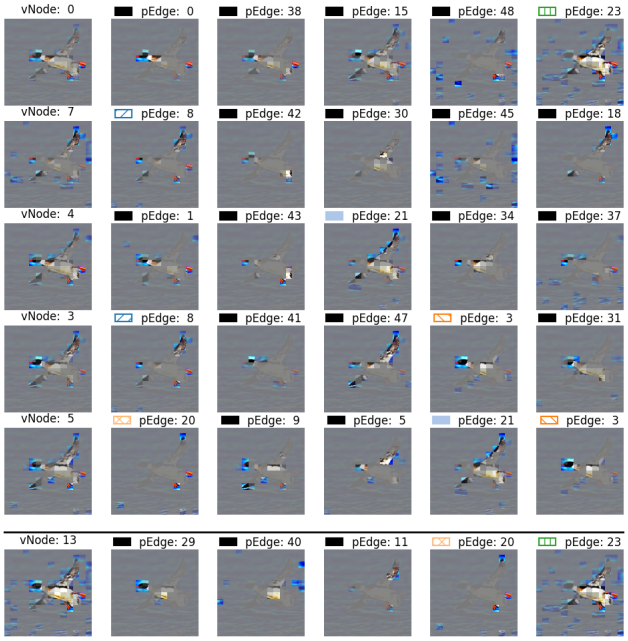


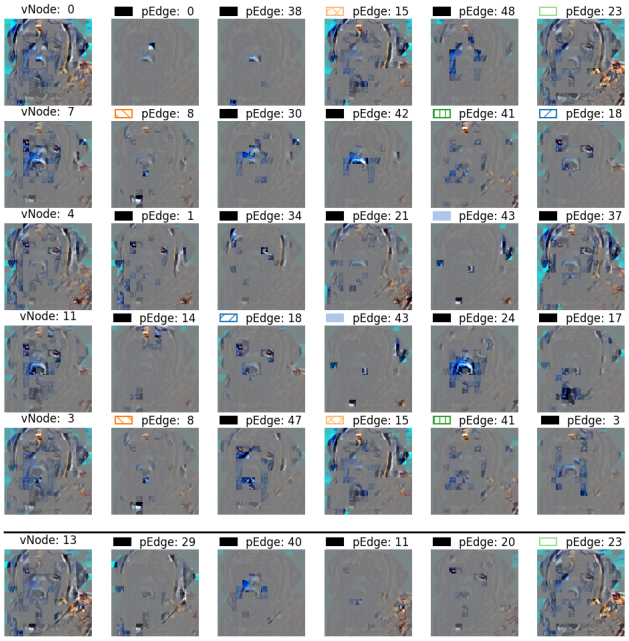
Figure 9. Example hypergraph structure used for visualization. Showing the four distinct feature types and the subset selection (top-1; root node) expert pooling used for classification - unused virtual edges are shown in light gray. Showing direct (0-hop; red) and indirect (1-hop; pink) virtual vertices, along with their membership primary hyperedges (orange), and the associated image vertices (blue). Features omitted in the graph visualizations are shown with open circles. Notably, a primary edge may be duplicated if it belongs to multiple virtual vertices.

intensity (no dimming for the highest contributions, maximal dimming for zero contributions). Finally, we add secondary virtual vertices linked to the primary hyperedges, further enriching each slice's representation by showing indirect (1-hop) influences. Fig. 9 provides a graphical depiction of this hierarchical structure, illustrating the direct and indirect virtual vertices and the connecting elements, indicating which components are plotted or excluded due to the pruned slice mechanism.

The following figures present graph visualizations that highlight the autosegmentation properties and hierarchical feature localization within the hypergraph structure, with distinct regions corresponding to features like eyes and feet. Notably, these visualizations are derived solely from the adjacency matrix rather than attention layers, though they exhibit structural properties similar to what one might expect from attention visualizations. This demonstrates that the adjacency relationships within the hypergraph capture meaningful spatial and semantic organization independently of the attention mechanisms.

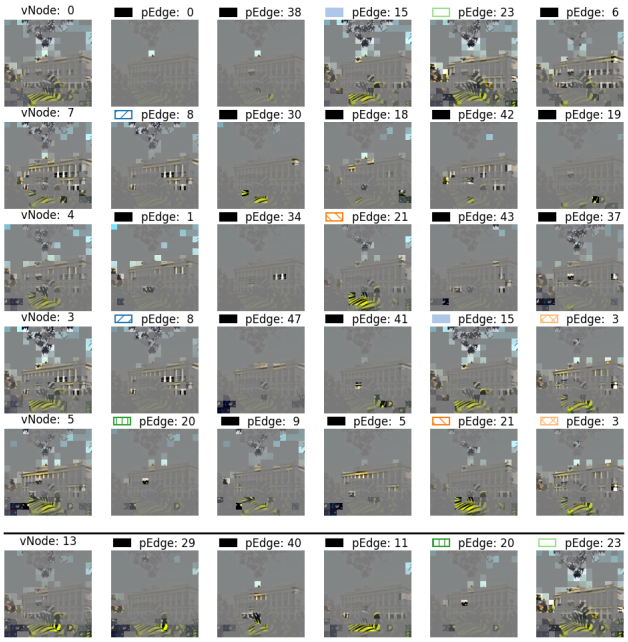


(a) Class label="Mergus serrator" (98).

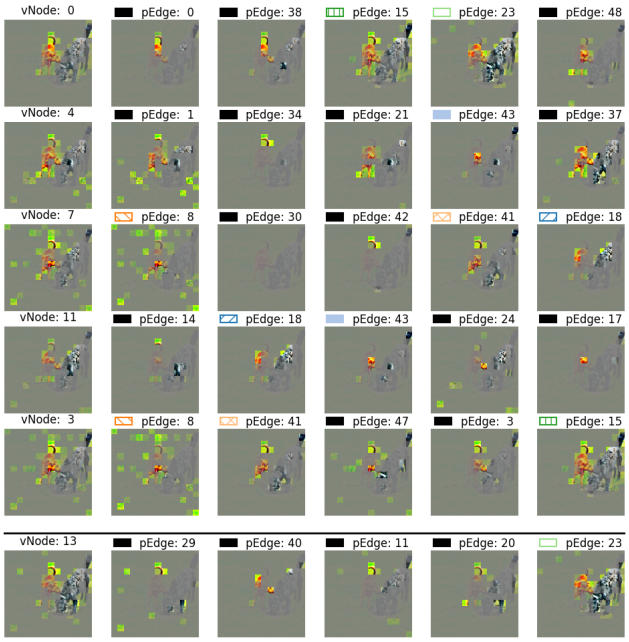


(b) Class label="Bull mastiff" (243).

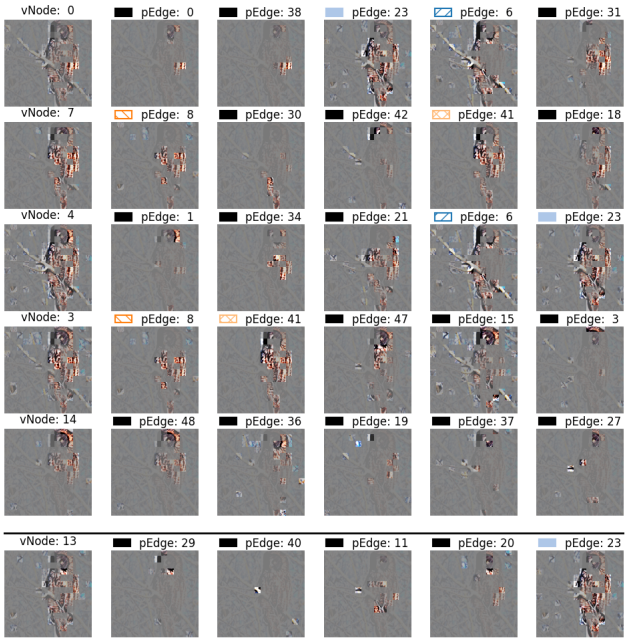
Figure 10. Graph visualizations from the HgVT-Ti model trained on ImageNet-1k, using samples from the ImageNet validation set. Showing top-5 direct virtual vertices and their top-5 highest contributing primary hyperedges above the horizontal line; top-1 indirect virtual vertex and its primary hyperedges below. Leftmost column shows aggregated summary of all primary hyperedges; remaining columns show individual primary hyperedges. Shared primary hyperedges are marked with unique identifier boxes; a black rectangle indicates no duplicates.



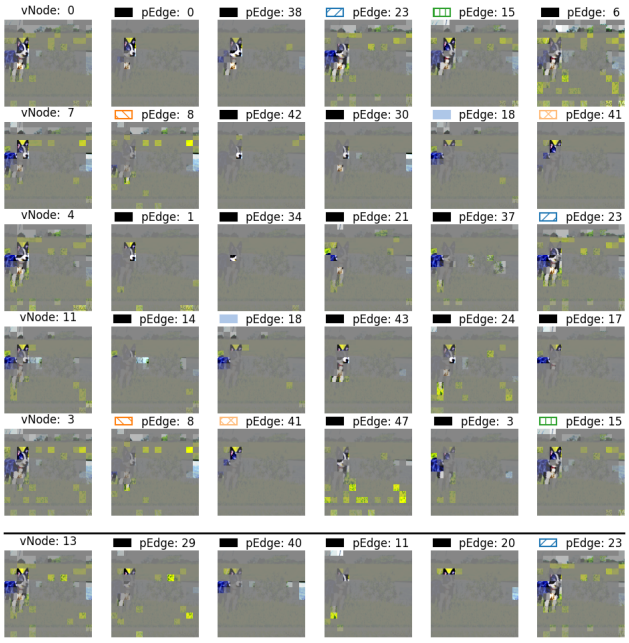
(a) Class label="Palace" (698).



(b) Class label="Irish terrier" (184).



(c) Class label="Great grey owl" (24).



(d) Class label="Border collie" (232).

Figure 11. Graph visualizations from the HgVT-Ti model trained on ImageNet-1k, using samples from the ImageNet validation set. Showing top-5 direct virtual vertices and their top-5 highest contributing primary hyperedges above the horizontal line; top-1 indirect virtual vertex and its primary hyperedges below. Leftmost column shows aggregated summary of all primary hyperedges; remaining columns show individual primary hyperedges. Shared primary hyperedges are marked with unique identifier boxes; a black rectangle indicates no duplicates.

F. Semantic Segmentation

In this section, we evaluate the performance of HgVT on the dense prediction task of semantic segmentation. Given the transformer backbone, we adopt the training protocol proposed in DINOv2 [38], which involves an initial finetuning phase at higher input resolutions on ImageNet-1k with positional embedding interpolation, followed by freezing the backbone and training segmentation heads. Final segmentation is then performed by merging overlapping "stencil" predictions at the segmentation training resolution (i.e. 512x512). Notably, freezing the backbone deviates from standard semantic segmentation training protocols. This is due to the fact that semantic segmentation relies heavily on spatial features (image vertices), and there is no straightforward gradient pathway for the hyperedge features, thereby preventing effective full-backbone finetuning.

F.1. Resolution Finetuning

To bridge the gap between pretraining and dense prediction tasks, we perform resolution finetuning, a process where the model is further trained on ImageNet-1k at a higher input resolution. While DINOv2 employs a resolution finetuning strategy at 416^2 for 10k steps using a cosine annealing learning rate schedule, we adopt a more lightweight approach inspired by TransNeXt [48]. Specifically, we finetune the model at a resolution of 384^2 for 5 epochs using a constant learning rate of 1e-5.

Additionally, to maintain consistent sparsity in the hypergraph representations at the higher resolution, we adjust the maximum population regularization value (β) to $|\mathcal{V}|/4$, where $|\mathcal{V}|$ is the number of vertices. This adjustment ensures that the model's structural regularization scales appropriately with the increased resolution. All other training hyperparameters (including data augmentation) remain identical to those used during the initial pretraining phase.

Table 10. Ablations on resolution finetuning HgVT-S.

Method	FT. Res.	Interp. PEs	Pop Max (β)	Top-1		Sparsity	
				224^2	512^2	224^2	512^2
Baseline	–	–	$1/6 \cdot \lvert\mathcal{V}_{224}\rvert$	81.20	78.59	0.637	0.750
A0	224^2	–	$1/4 \cdot \lvert\mathcal{V}_{224}\rvert$	80.95	77.84	0.603	0.721
B0	384^2	✗	$1/6 \cdot \lvert\mathcal{V}_{224}\rvert$	78.24	81.13	0.875	0.951
B1	384^2	✓	$1/6 \cdot \lvert\mathcal{V}_{224}\rvert$	78.84	81.54	0.875	0.951
C0	384^2	✓	$1/6 \cdot \lvert\mathcal{V}_{384}\rvert$	80.03	82.26	0.611	0.764
C1	384^2	✓	$1/4 \cdot \lvert\mathcal{V}_{384}\rvert$	80.11	82.32	0.592	0.743
C2	384^2	✓	$1/3 \cdot \lvert\mathcal{V}_{384}\rvert$	80.01	82.28	0.619	0.771

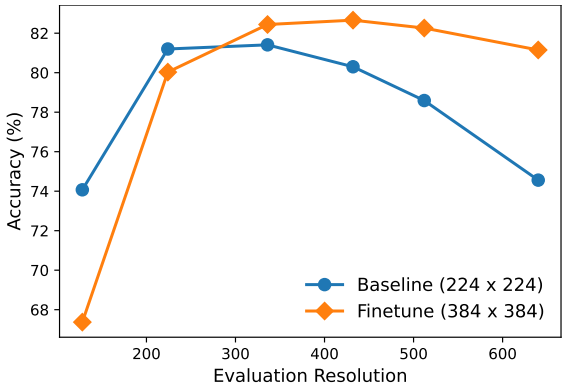


Figure 12. Top-1 ImageNet-1k accuracy for HgVT-S before and after resolution finetuning.

In Tab. 10, we present an ablation study evaluating the impact of interpolating versus reinitializing positional embeddings, as well as the effect of varying the maximum population regularization value (β). We find that interpolating positional embeddings leads to better performance, while increasing β helps prevent over-sparsification, with $\beta = \frac{1}{4}|\mathcal{V}|$ yielding the best results at higher resolutions. Interestingly, this setting slightly degrades performance when maintaining the original training resolution, suggesting that the benefits of a larger population regularization are resolution-dependent. Fig. 12 shows the Top-1 ImageNet accuracy across resolutions for the baseline HgVT-S and method C1, revealing trends that are remarkably consistent with the resolution finetuning behavior observed in DINOv2 [38].

F.2. Segmentation Results

Following the finetuning phase, we train segmentation heads on top of the frozen backbone, following the protocol used by DINOv2, with training hyperparameters summarized in Tab. 11. We evaluate performance on the ADE20k [67], CityScapes [6], and PASCAL VOC [13] datasets. To better understand the feature representations learned by HgVT, we compare the L2 feature norms of the last four layers of HgVT-S and DINOv2-S for an example image, as shown in Fig. 13. Notably, HgVT exhibits significantly sparser feature activations compared to DINOv2. This suggests that relying solely on the final feature layer may limit segmentation performance, leaving gaps in otherwise contiguous regions.

Given the uncertain behavior of the sparse feature activations, we explore several segmentation head architectures to assess the effectiveness of each in decoding the sparse image vertex features.

- **Linear Head:** A simple linear projection following a batch normalization layer as used in DINOv2.
- **MLP Head:** A two-layer perceptron following Linear-BN-SiLU-Linear.

Table 11. Segmentation Hyperparameters.

Parameter	Value
Train Resolution	512×512
Global Batch Size	16
Schedule	Poly
Power	1.0
Total Steps	40k
Warmup Steps	1.5k
Optimizer	AdamW
Peak LR	1e-3
Weight Decay	1e-4
(β_1, β_2)	(0.9, 0.999)
Resize Ratio	$0.5 - 2.0$
Augmentations	Random Crop, Flip, Photometric

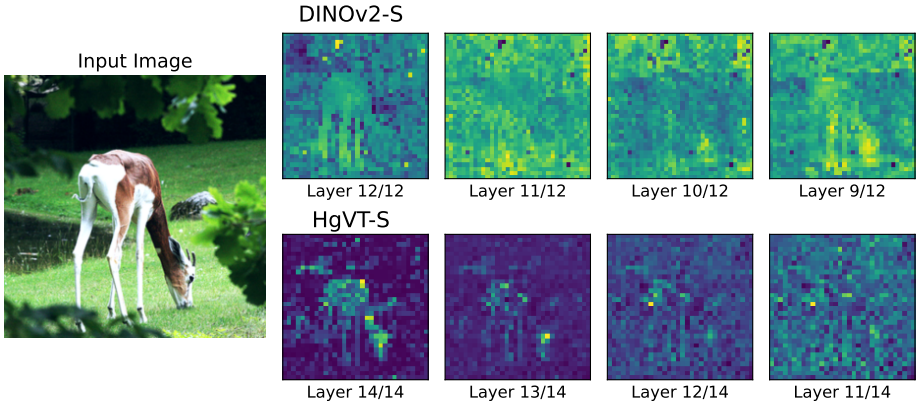


Figure 13. Comparison of DINOv2-S (top) and HgVT-S (bottom) spatial features for the last 4 layers in each network. Plotting the per-token L2 norm to visualize the HgVT feature sparsity.

- **Conv-MLP Head:** Similar to the MLP head but with a 3x3 convolution as the input layer.
- **Pyramid Pooling Module (PPM):** Module proposed by PSPNet [65], which utilizes multi-level pooling for isotropic input features.
- **Upsampled PPM Head (PPMU):** An enhanced PPM implementation which uses a 2x up-sampling step with pixel shuffle before the final MLP.

Table 12. Semantic Segmentation results on ADE20k using the frozen HgVT-S backbone. Head method includes input configuration: `-1` last backbone layer only, `-4` last four backbone layers concatenated. Showing mIoU (%) and Pixel Accuracy (%) where available. *Our evaluation. †Results from `github.com/CSAILVision/semantic-segmentation-pytorch`. ‡Results from DINOv2 [38].

Backbone			Head			ADE20k		CityScapes		PASCAL VOC	
Method	Size	Frozen	Method	Size	Multiscale	mIoU	Acc.	mIoU	Acc.	mIoU	Acc.
Swin-Ti [33]	28.3M	✗	UperNet [59]	60M	✓	46.1	–	–	–	–	–
TransNeXt-Ti [48]	28.2M	✗	UperNet [59]	59M	✗	51.1	–	–	–	–	–
TransNeXt-Ti [48]	28.2M	✗	UperNet [59]	59M	✓	51.7	–	–	–	–	–
TransNeXt-Ti [48]	28.2M	✗	Mask2Former [4]	47.5M	✗	53.4	–	–	–	–	–
ResNet-18 [20]	11.5M	✗	PPM-1	12.9M	✗	33.8†	76.1†	–	–	–	–
ResNet-50 [20]	25.6M	✗	PPM-1	23.2M	✗	41.3†	79.7†	–	–	–	–
ResNet-101 [20]	44.5M	✗	PPM-1	23.2M	✗	42.2†	80.6†	78.4	–	82.6	–
DINOv2-S/14 [38]	22.1M	✓	Linear-1	59.3k	✗	44.3	79.5*	66.6	–	81.1	95.9*
DINOv2-S/14 [38]	22.1M	✓	Linear-4	237k	✗	46.0*	80.1*	–	–	81.8*	96.0*
DINOv2-S/14 [38]	22.1M	✓	Linear-4	237k	✓	47.2	–	77.1	–	82.6	–
DINOv2-G/14 [38]	1.10B	✓	Linear-1	237k	✗	49.0	–	71.3	–	83.0	–
OpenCLIP-G/14 [25]	1.01B	✓	Linear-1	214k	✗	39.3‡	–	60.3‡	–	71.4‡	–
HgVT-S/16	22.9M	✓	Linear-1	34.6k	✗	12.0	43.3	30.2	72.9	34.0	81.4
HgVT-S/16	22.9M	✓	Linear-4	138k	✗	26.7	68.5	52.4	89.3	66.7	91.7
HgVT-S/16	22.9M	✓	MLP-4	235k	✗	28.5	71.8	58.0	91.7	72.9	93.6
HgVT-S/16	22.9M	✓	ConvMLP-4	1.84M	✗	33.5	74.3	64.5	93.1	76.1	94.4
HgVT-S/16	22.9M	✓	PPM-4	15.5M	✗	36.0	75.7	68.0	93.8	77.9	94.9
HgVT-S/16	22.9M	✓	PPMU-4	17.4M	✗	37.6	76.4	69.8	94.3	79.0	95.1

Segmentation results are shown in Tab. 12. Consistent with the feature norm analysis, the Linear Head underperforms, particularly when applied solely to the final feature layer. To investigate this further, we also evaluate a linear head that combines features from the last four backbone layers (consistent with the multiscale method in DINOv2). While this approach improves performance compared to using only the final layer, it still falls short of more complex architectures. This suggests that deeper features mitigate some of the sparsity effects observed in Fig. 13, while linear projections alone are insufficient for fully decoding the hypergraph representations. While the more complex PPMU method achieves an mIoU of 37.6% on ADE20K, it falls short of both DINOv2-S and state-of-the-art methods.

In contrast, results on CityScapes and PASCAL VOC are stronger, with the PPMU heads closing the gap on PASCAL VOC

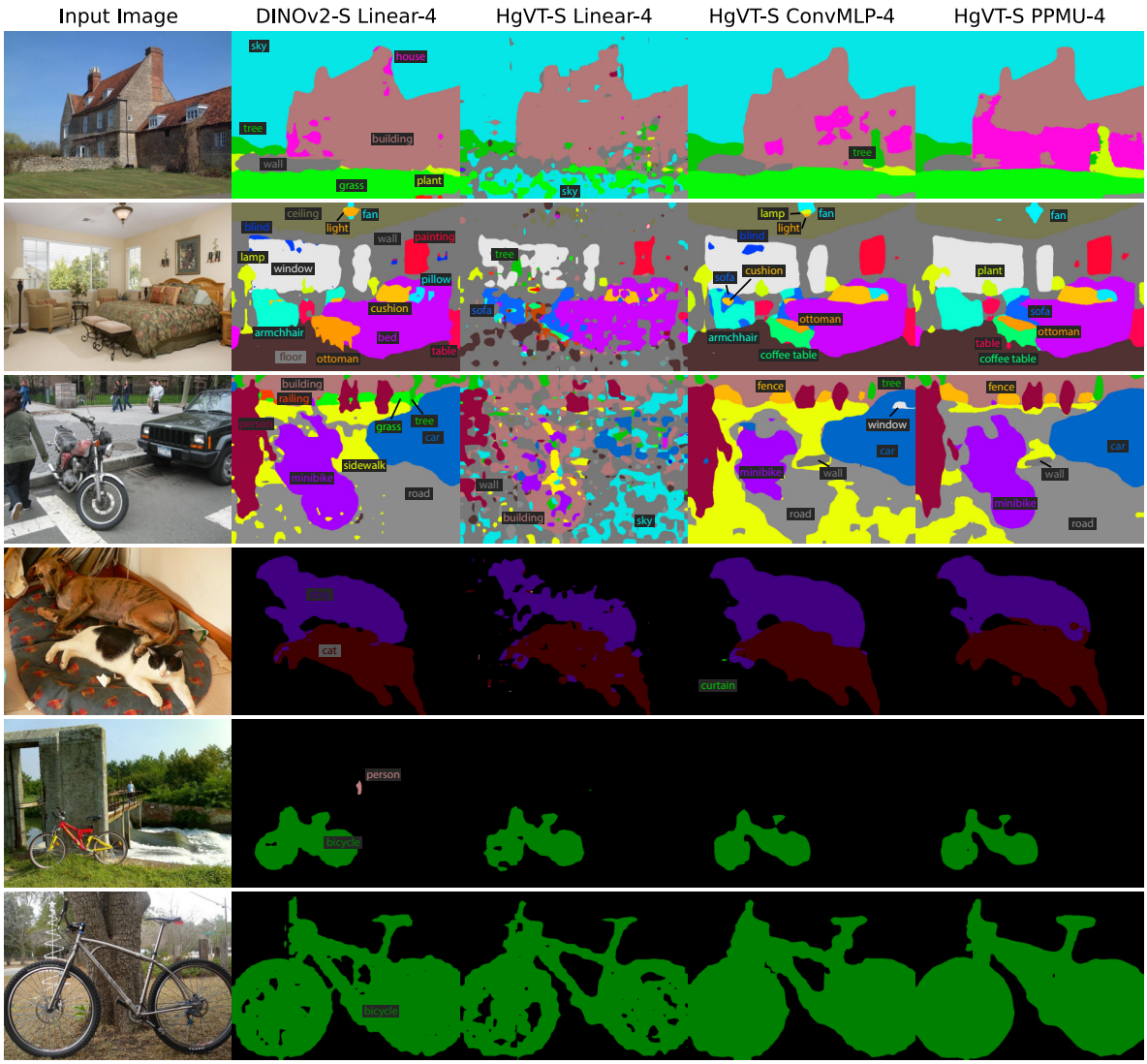


Figure 14. Semantic Segmentation Visualization. Showing examples from ADE20k (top) and PASCAL VOC (bottom).

and surpassing DINOv2 Linear-1 classifier on CityScapes. Notably, all convolution-based methods outperform OpenCLIP-G on these two datasets, suggesting that (1) the poor ADE20K results are partially attributable to class confusion and (2) the sparse features result in discontinuous regions, which degrade segmentation performance. The convolution-based methods help smooth out these discontinuities, improving overall performance. Additionally, the reduced class count (20 vs. 150) likely mitigates class confusion, contributing to stronger performance on CityScapes and PASCAL VOC.

We attribute this low performance to several factors. First, the lack of backbone fine-tuning leads to object class confusion, where similar classes (e.g., cushion and pillow) that were not targeted during ImageNet-1k training are incorrectly assigned. Second, the high degree of feature sparsity encouraged by population regularization may result in localization errors, where objects are not encoded at the correct pixel location. As supporting evidence, we measure a 4.3% lower mIoU and 2.7% lower pixel accuracy on ADE20k when using a Linear-4 head with configuration B1 in Tab. 10. Third, the patch size of 16×16 pixels further reduces segmentation localization compared to the more commonly used 8×8 down-sampling. Notably, DINOv2 uses a 14×14 patch size, self-supervised learning, and ImageNet-22k pretraining, resulting in denser features (see Fig. 13), which likely accounts for part of the performance gap. Finally, a large amount of information –including the hyperedge features and virtual nodes – is not directly used in semantic prediction due to the lack of direct spatial alignment. Leveraging these additional features may improve boundary detection and class distinction, highlighting areas for future exploration.

The segmentation visualizations in Fig. 14 align with these findings. The linear head on HgVT produces discontinuous segmentation regions, whereas the convolution-based methods help fill these gaps. The PPMU head appears to over-smooth

the results, leading to missed fine details (e.g., the bedroom and bicycle in the second and last rows). In certain PASCAL VOC examples, HgVT with a linear head outperforms DINOv2, where increased sparsity results in more well-defined segmentation regions (e.g., the large bicycle image in the last row). Finally, class confusion can be seen in the bedroom scene (second row), where the top of an ottoman is correctly identified while the bottom is misclassified as a coffee table. This supports our hypothesis that object class confusion is occurring and may partially explain the poor ADE20K performance.

F.3. Using Semantic Segmentation for Interpretability

Aside from benchmark evaluation, the linear segmentation results also provide insight into how the model encodes information. Large contiguous regions are sparsely represented by the correct class, with gaps filled by high-frequency or default classes (e.g., wall and sky). This suggests that the model assigns the correct class to a small subset of vertices, efficiently summarizing the local structure rather than encoding them uniformly. The hypergraph visualization results in Appendix E support this interpretation, showing that regions like water, grass, and sky are not contiguously covered but instead exhibit sparse coverage. This pattern may be analogous to a dithering effect used to represent continuous shading with binary values. A convolution operation would efficiently reconstruct the full structure by locally propagating this summarized information, which is supported by the ConvMLP results.

G. Image Retrieval

This section expands upon the image retrieval description in the main paper to provide additional implementation details and supporting evidence. Our image retrieval framework is structured around two primary first-pass search methods: pooled similarity and volumetric similarity. Both methods leverage the pooled embedding, which serves as the input to the classifier head and integrates both pooled image features and expert edge features.

- **Pooled Similarity (PS):** This method computes similarity scores by comparing the pooled embeddings through a cosine similarity metric. The pooled embedding serves as a generalized representation of each image, aligning with standard vector-based similarity searches, making it both effective and efficient as a first-pass retrieval approach.
- **Volumetric Similarity (VS):** Unlike pooled similarity, volumetric similarity incorporates the hypergraph structure by treating the pooled embedding as a centroid. Similarity is determined using an approximate Mahalanobis distance, which accounts for the distributional spread around the centroid based on a subset of primary hyperedges. This approach captures overlap with less prominent, yet relevant, features, enabling a spatially-aware similarity measure that aligns more closely with nuanced structural characteristics.

Individually, both methods perform effectively as first-pass search strategies; however, to further harness the structure of the hypergraph, we introduce an adaptive reranking phase. This phase refines retrieval results by re-evaluating similarity across a short list of top R candidates, using a more detailed hypergraph similarity measure. The adaptive reranking can be applied to each of the first-pass methods, resulting in Adaptive Volumetric Similarity (AVS) and Adaptive Pooled Similarity (APS). By capitalizing on the hierarchical and relational information embedded within the hypergraph, these adaptive methods enhance retrieval precision beyond the initial search.

G.1. Graph Pruning

For methods that leverage the hypergraph structure, we employ a pruned graph representation based on primary hyperedge features ($p\mathcal{E}$). The pruning process begins by selecting the top-1 expert edge as the root, which serves as the initial focus for identifying key structural components. From this root, we identify the top M virtual vertices that contribute most significantly to the expert edge. For each of these M virtual vertices, we further select the top N primary hyperedges connected to it. This yields a total of $M \times N$ hyperedge features, where we choose $M = 3$, $N = 4$, and $M \times N = 12$ to prove a balanced between representation coverage and computational efficiency. Finally, the selected hyperedge features are deduplicated and ranked based on their overall contribution to the final prediction. Notably, this process is very similar to the slice visualization described in Appendix E, and illustrated in Fig. 9.

G.2. Volumetric Similarity

Volumetric similarity leverages the hypergraph structure by treating the pooled embedding x of each image as a centroid, with the pruned primary hyperedges defining a spread around this centroid. Each of the two distributions can then be represented by a centroid and covariance matrix, (x_1, Σ_1) and (x_2, Σ_2). We then quantify the similarity between these distributions using the Mahalanobis distance with a combined covariance matrix, capturing both the central positions and spreads of the distributions

to measure their overlap.

$$d_M(x_1, x_2)^2 = \sqrt{(x_1 - x_2)^T \left(\frac{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}{2}\right)^{-1} (x_1 - x_2)} \tag{26}$$

where $\mathbf{\Sigma} = \frac{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}{2}$ is the average covariance matrix between the two distributions. To reduce computational complexity, we approximate $\mathbf{\Sigma}$ as a diagonal matrix, assuming minimal covariance between features. Each feature's combined variance simplifies to $\sigma^2 = (\sigma_1^2 + \sigma_2^2)/2$, yielding:

$$d_M(x_1, x_2)^2 \approx \sum_i \frac{(x_{1,i} - x_{2,i})^2}{\sigma_i^2} \tag{27}$$

where σ_i^2 represents the average variance of the i-th feature across the two distributions.

While the diagonal approximation reduces complexity, calculating $1/\sigma_i^2$ for each feature remains computationally demanding. To further optimize, we approximate each variance term $\sigma_i^2 \approx \bar{\sigma}^2 + \delta_i^2$, where $\bar{\sigma}^2$ is the mean variance across features, and δ_i^2 represents the deviation from this mean. Finally, we can then express $1/\sigma_i^2$ using a Taylor series expansion:

$$\frac{1}{\sigma_i^2} \approx \frac{1}{\bar{\sigma}^2}(1 - \eta_i + \eta_i^2 + \dots), \quad \eta_i = \frac{\delta_i^2}{\bar{\sigma}^2} \tag{28}$$

By truncating this expansion after the first few terms, we achieve an efficient approximation for the Mahalanobis, requiring at most a single division per comparison:

$$d_M(x_1, x_2)^2 \approx \rho \sum_i (x_{1,i} - x_{2,i})^2 (1 - (\rho \cdot \delta_i^2) + (\rho \cdot \delta_i^2)^2), \quad \rho = \frac{1}{\bar{\sigma}^2} \tag{29}$$

This approach allows for efficient computation that remains relatively close to the simpler cosine similarity measure, while also capturing greater variance introduced by the hypergraph structure.

In practical terms, while these approximations do not hold universally, the deviation is small enough that the simplified form remains effective for our retrieval framework. When truncating the Taylor series to the first-order approximation the term $(1 - \rho \cdot \delta_i^2)$ must be clamped to a positive value, as large deviations in certain elements can cause this term to become negative, violating the mathematical definition of variance. Notably, this clamping is unnecessary for the second-order approximation, where additional terms sufficiently stabilize the variance without requiring this constraint.

G.3. Adaptive Reranking

The adaptive reranking process refines the initial retrieval results by re-evaluating a short list of top R entries selected through one of the first-pass similarity methods. For each of these R entries, we perform a graph-based similarity search, focusing on the pruned primary hyperedge features of each graph. The similarity is computed as the average distance between corresponding primary hyperedges in the query and candidate graphs.

While effective, this approach can be computationally expensive, requiring $\mathcal{O}(R \cdot (M \times N)^2)$ operations, where M and N represent the number of virtual vertices and primary hyperedges, respectively. However, the diversity regularization applied during training ensures minimal overlap between comparisons, resulting in a sparse correlation matrix with mostly zero similarities. This sparsity leads to redundant computations, making the process well-suited for optimization through hash-based acceleration.

To take advantage of this sparsity, we employ a centroid-based hashing mechanism, which reduces the number of necessary comparisons. Specifically, we learn a set of H centroids that define H distinct bins, with each primary hyperedge feature in the pruned graphs (both query and candidate) hashed into these bins. By limiting comparisons to features within the same bin, and only considering the top C most relevant comparisons (defined by the query graph), we can reduce the overall complexity to $\mathcal{O}(R \cdot C)$. This approach enables adaptive reranking to achieve higher precision with significantly reduced computational costs, leveraging the sparse structure introduced by hypergraph regularization.

G.4. Centroid Hashing

To implement the hashing mechanism described in adaptive reranking, we learn a set of H centroids that define bins for efficient similarity comparisons. Empirically, we find that setting $H = 10$ provides effective separation when $M \times N = 12$, balancing coverage with computational efficiency.

These centroids are trained using the Adam optimizer over a dataset created from all pruned primary hyperedge features across the test dataset. The optimization objective involves minimizing the distance to the closest centroid while maximizing the distance to all other centroids, thereby ensuring distinct and well-separated bins. Additionally, we incorporate the same density regularization term applied to the expert edges, promoting a broader feature spread within each centroid bin. The combined loss function is thus:

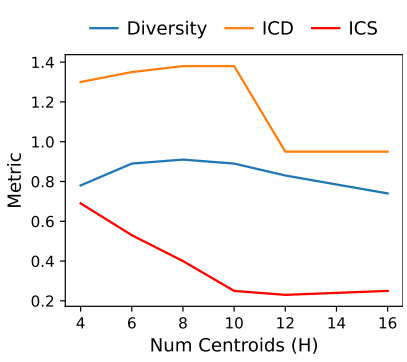
$$\mathcal{L}_{\text{centroid}} = ||y - c_{n1}||^2 - \lambda_{ICD} \cdot ||y - c_{n2}||^2 + \lambda_{DEN} \cdot \text{den}(c_{n1}) \tag{30}$$

where y is the input feature vector, c_{n1} and c_{n2} are the nearest and second nearest centroids, λ is a loss weight factor, and $\text{den}(\cdot)$ is the density regularization term computed over the batch. This objective minimizes the distance of each feature to its nearest centroid, while enforcing a margin with the second-closest centroid. The regularization term further ensures that centroids remain well-utilized across the feature space.

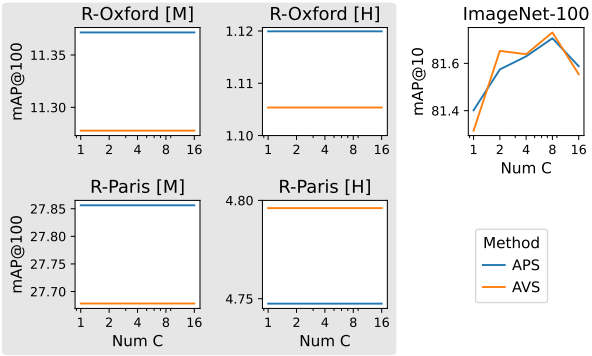
Emperically, we find that a learning rate of 4×10^{-3} works well for a batch size of 512, setting $\lambda_{ICD} = 0.1$ and $\lambda_{DEN} = 0.5$. In practice, centroid training converges rapidly, requiring only two epochs on larger datasets such as ImageNet and CIFAR. For smaller datasets, such as Oxford and Paris, training requires approximately eight epochs.

G.5. Retrieval Hyperparameter Ablations

We evaluate the influence of four critical hyperparameters on retrieval performance: the number of centroids H, the number of graph similarity comparisons C, the Mahalanobis approximation order, and the shortlist rank R used in adaptive reranking. Results for H and C are presented in Fig. 15, while Fig. 16 highlights the effects of the Mahalanobis approximation order and shortlist rank.



(a) Hash Centroid Scaling.



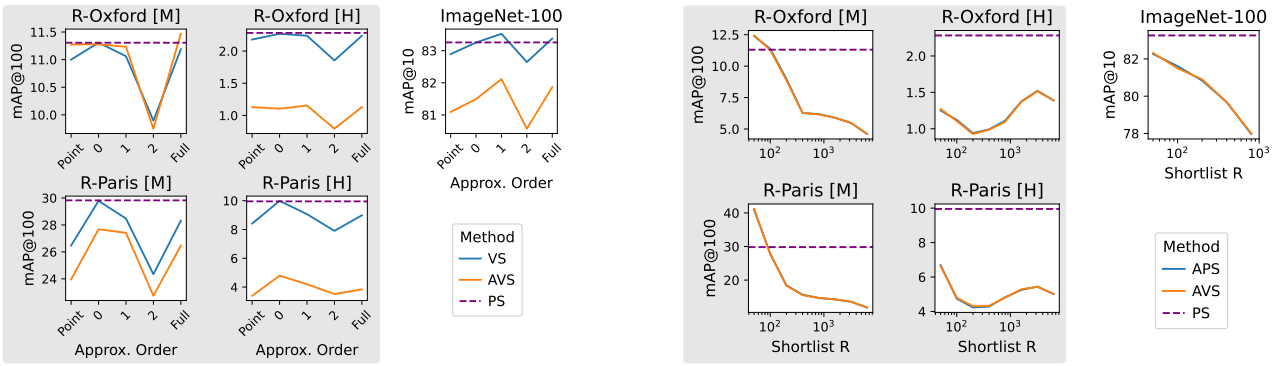
(b) Graph similarity Comparison Scaling.

Figure 15. Hyperparameter scaling behavior for adaptive re-rank method. (a) impact of hash bin clustering metrics as a function of centroid count on CIFAR-100; (b) retrieval performance as a function of graph similarity comparisons for: (left) Oxford and Paris (right) and KNN retrieval on ImageNet-100 with HgVT-Lt. Notably, Oxford and Paris are insensitive, likely due to reduced feature diversity from landmarks.

Effect of Centroid Count: Fig. 15a presents the relationship between H and final centroid training metrics. Namely we use a diversity measure (1.0 represents uniform distribution among centroids), inter-cluster distance (ICD), and intra-cluster similarity (ICS) for the HgVT-Mu model trained on CIFAR-100. Increasing H improves all metrics up to a point, followed by a degradation due over granularization. We find that $H = 10$ achieves the best result for the chosen graph configuration ($N = 3$ virtual vertices, $M = 4$ primary hyperedges), with a notable drop in ICD at $H \geq 12 = N \times M$. This choice allows the bins to remain distinct enough to provide adequate separation, while also providing sufficient overlap with an expectation value of 1.2 hyperedges per bin.

Effect of Comparison Count: Fig. 15b illustrates the performance of varying C for the HgVT-Lt model, trained on ImageNet-100, across different retrieval benchmarks. While the Oxford and Paris datasets exhibit insensitivity to C, potentially due to their dependence on salient features emphasized by the diverse ImageNet-100 set, ImageNet-100 retrieval performance peaks at $C = 8$. For computational efficiency, $C = 4$ is selected as a trade off, maintaining comparable mAP@10 performance while requiring fewer similarity comparisons.

Effect of Mahalanobis Approximation Order: Fig. 16a examines the impact of the Mahalanobis approximation order on volumetric similarity performance. Several configurations are evaluated, including point-wise approximation (where the query variance is set to 0 and the full candidate variance is precomputed as $1/\sigma_i^2$), as well as 0th, 1st, and 2nd order



(a) Mahalanobis Order.

(b) Re-rank short-list size.

Figure 16. Hyperparameter scaling behavior for adaptive re-rank methods with HgVT-Lt trained on ImageNet-100. (a) impact of Mahalanobis approximation order, showing point-wise, 0th, 1st, 2nd order, and full ($N = \infty$); (b) impact of short-list size R on metrics. In both figures: (left) mAP retrieval on Oxford and Paris (right) and KNN retrieval for ImageNet-100. Also showing baseline using pooled similarity (PS) as horizontal purple dashed line.

Taylor series approximations, and the full computation of $1/(\sigma_{1,i}^2 + \sigma_{2,i}^2)$. Results indicate that the 0th order approximation consistently achieves the best performance, balancing accuracy and efficiency by leveraging only $\bar{\sigma}^2$. Conversely, the 2nd order approximation fails across all cases, likely due to instability from the $(\delta_i^2)^2$ term becoming larger than 1, causing the approximation to break down. These findings suggest that the simpler 0th order approach is both effective and computationally optimal for volumetric similarity.

Effect of Shortlist Size: Fig. 16b explores the effect of shortlist size R on adaptive metric performance. Across all methods, performance degrades as R increases, driven by confusion in the graph similarity metric, which becomes more susceptible to distraction by sub-salient features. Despite this trend, a shortlist size of $R = 100$ strikes a suitable balance, limiting significant distractions while maintaining enough candidates to sufficiently approximate the full mAP metric, which favors smaller k-rank evaluations (mAP@k).

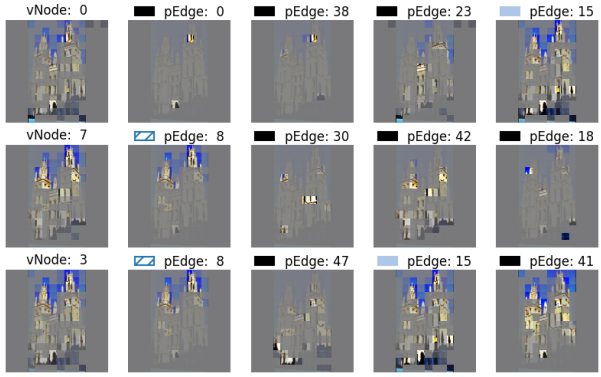
G.6. Visualizing Adaptive Reranking

This section provides a visual analysis of the adaptive reranking process using the Oxford dataset, demonstrating how structural similarities in hypergraphs influence retrieval precision.

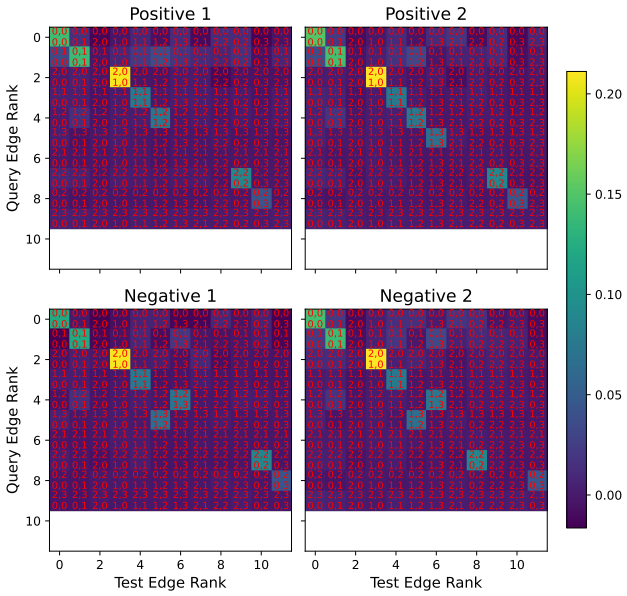
In Fig. 17, we present a test query image alongside two known positive images and two known negative images. For each of these five images, we show the pruned hypergraph visualizations, including similarity scores for each of the primary hyperedges. Notably, distinct structural patterns emerge in the similarity scores, with higher scores between the query and positive images compared to the negative images. Additionally, we observe that the query edge rank in this example stops at 9, while the test edge ranks extend to 12. This discrepancy arises because two of the primary hyperedges in the pruned query hypergraph are duplicates, removed during deduplication, resulting in a total of 10 unique hyperedges.

Fig. 18 further illustrates the impact of adaptive reranking on retrieval quality for the Oxford Medium dataset. Using the same query image from Fig. 17, we first display the top $R = 100$ images retrieved based on pooled similarity ranking. In this initial retrieval, positive images are dispersed throughout the ranks, and several irrelevant images, including those without buildings, appear near the top. Applying adaptive reranking significantly improves the results: positive images are shifted to higher ranks, while irrelevant images are moved toward the end of the list. This visual evidence highlights the effectiveness of adaptive reranking in refining retrieval results by leveraging hypergraph structural information to enhance semantic alignment.

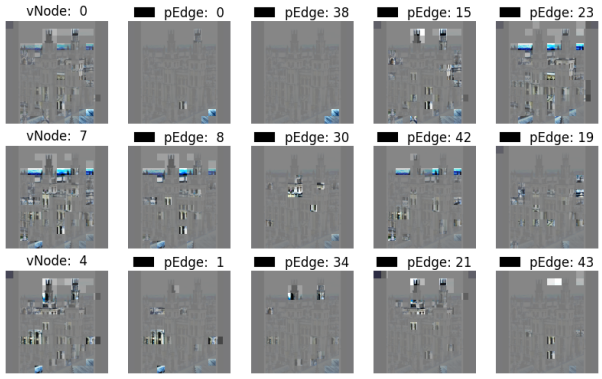
(a) Query and Test Images.



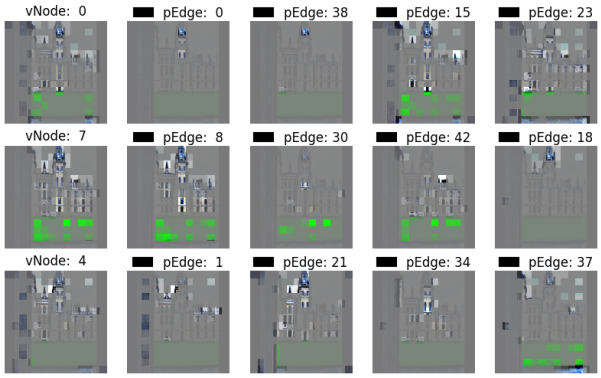
(b) Pruned Query Hypergraph.



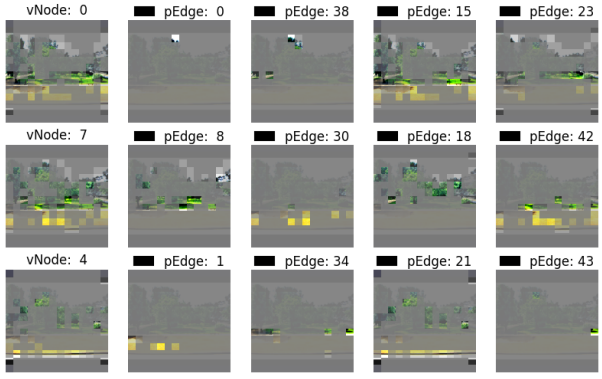
(c) Aggregate Feature Similarities.



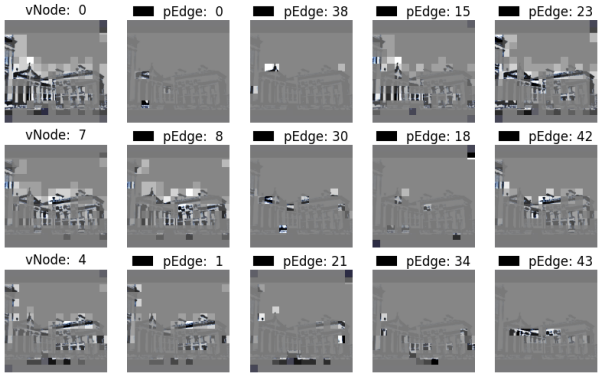
(d) Pruned Positive 1 Hypergraph.



(e) Pruned Positive 2 Hypergraph.



(f) Pruned Negative 1 Hypergraph.



(g) Pruned Positive 2 Hypergraph.

Figure 17. Example Revisted Oxford retrieval for query (Ashmolean Museum), with two positive and two negative results for HgVT-Ti. (a) showing input images, (b) pruned hypergraph visualization for the query image, (c) aggregate hyperedge similarity scores, (d-e) pruned hypergraph visualizations of the positive image pairs, (f-g) pruned hypergraph visualizations of the negative image pairs. All hypergraph visualizations label the top-3 virtual vertices (vNode) and their corresponding top-4 primary hyperedges (pEdge). If a primary hyperedge connects to multiple virtual vertices, this link is indicated by a unique marker other than solid black. In (c), the corresponding query (top) and test hyperedge (bottom) coordinates are indicated by red numbers: as `vNode,pEdge`. For example: pEdge 47 in the query hypergraph would be `2,1`. In all cases, query pEdge 8 (`2,0`) has the highest similarity with pEdge 8 (`1,0`) in the test images.

(a) Pooled Similarity Ranking.

(b) Adaptive Pooled Similarity Ranking.

Figure 18. Top-100 ranking for the medium split of the query in Fig. 17 using HgVT-Ti. Showing (a) the results using pooled similarity ranking, (b) after re-ranking the top-100 shortlist using pruned hypergraph similarity. Positive matches are shown with thick cyan boarders, while negative matches use red boarders. The rank position is indicated by a number in the upper left corner of each image.

H. Additional Ablations

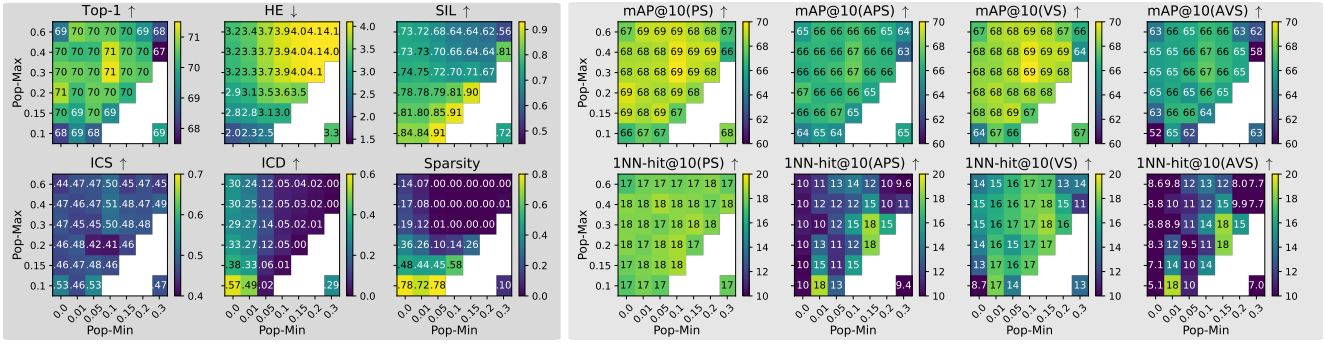
This section evaluates the design choices and hyperparameters shaping the performance and efficiency of the HgVT models. The primary ablations are conducted on HgVT-Lt, trained on ImageNet-100, to analyze architectural trade-offs between accuracy, computational cost, and model size. To explore the impact of population regularization hyperparameters more comprehensively, we utilize a smaller model, HgVT-Mu, trained on CIFAR-100 (details in Appendix I). This allows for detailed hyperparameter sweeps to assess their effects on graph quality, sparsity, retrieval performance, and inter-metric correlations. Additionally, we investigate the influence of expert pooling regularization parameters using HgVT-Mu to better understand their role in balancing sparsity and performance. Insights from these evaluations guide the selection of optimal configurations and provide a deeper understanding of the underlying model behavior.

H.1. Population Regularization Sweeps

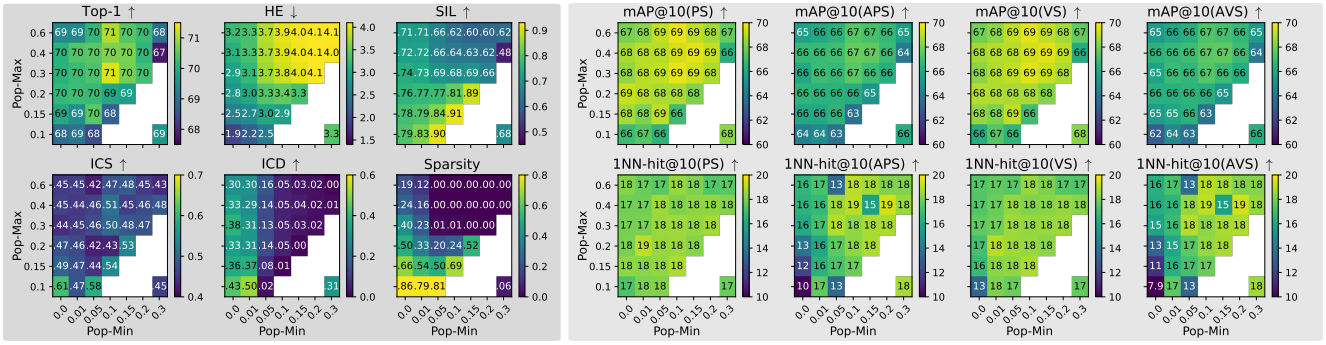
The population regularization mechanism facilitates learned self-sparsification and clustering within the generated hypergraphs. It is defined by the population regularization minimum density (γ) and maximum density (β), with the regularization terms encouraging soft adjacency membership contributions to remain within these bounds. A sweep of these parameters, normalized to the vertex count $|\mathcal{V}|$ is presented in Fig. 19 for the HgVT-Mu model, comparing the standard Hadamard edge attention modulation to the modified Hadamard edge attention modulation.

The results in Fig. 19 indicate that the modified Hadamard modulation consistently outperforms the standard approach. This improvement aligns with expectations, as the modified modulation removes the positive influence of non-membership vertices, thereby enhancing the accuracy of edge relationships. Both parameter grids form distinct performance landscapes, revealing regions where over-sparsification occurs and others where structural collapse leads to a maximally connected graph (sparsity = 0). Interestingly, top-1 accuracy and retrieval metrics generally favor the maximally connected case initially, but performance begins to degrade beyond a certain point. This suggests that the metrics benefit from a weakly maximally connected graph – characterized by softer membership weights – over a strongly maximally connected graph with more rigid weights. However, while a maximally connected structure may boost certain metrics temporarily, it ultimately hinders precise structural extraction and efficient computation, both of which rely on maintaining an appropriate level of sparsity.

To validate the findings from the HgVT-Mu model at scale, Fig. 20 presents a similar population regularization analysis for the HgVT-Lt model, trained on ImageNet-100. This analysis uses a coarser parameter grid and focuses on Top-1 accuracy and graph quality metrics. The results demonstrate a similar performance pattern to that observed with HgVT-Mu, but with a



(a) Standard Hadamard Soft Membership Modulation.



(b) Modified Hadamard Soft Membership Modulation.

Figure 19. Effect of population regularization minimum (γ) and maximum (β) density limits, for CIFAR-100. Regularization is normalized by $|\mathcal{V}|$, such that 1.0 corresponds to $\beta, \gamma = |\mathcal{V}|$. Lower-right cell in each subplot represents population regularization disabled. Left, showing top-1 accuracy and graph quality metrics hyperedge entropy (HE), silhouette score (SIL), intra-cluster similarity (ICS), inter-cluster distance (ICD), and sparsity (spA). Right, showing mAP@10 for image retrieval and top-10 hit-rate with top-1 CLIP-B ranking for four retrieval methods: standard, adaptive (A), volumetric overlap (V), and adaptive volumetric (VA). Further comparing (a) with standard Hadamard (bounded between 0 and 1), and (b) with modified Hadamard (bounded between -1 and 1) modulation in edge attention.

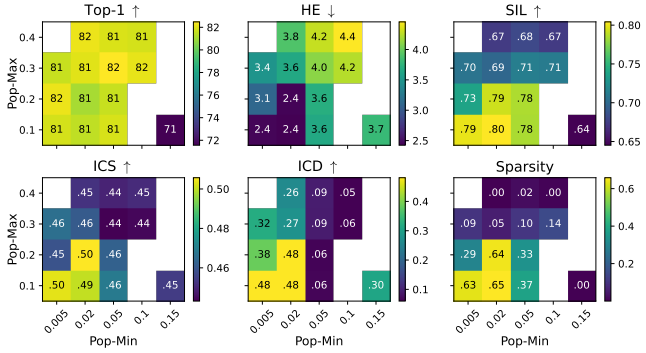


Figure 20. Effect of population regularization minimum (γ) and maximum (β) density limits, for ImageNet-100. Regularization is normalized by $|\mathcal{V}|$, such that 1.0 corresponds to $\beta, \gamma = |\mathcal{V}|$. Lower-right cell in each subplot represents population regularization disabled. Showing top-1 accuracy and graph quality metrics hyperedge entropy (HE), silhouette score (SIL), intra-cluster similarity (ICS), inter-cluster distance (ICD), and sparsity (spA).

noticeable shift toward lower values of the normalized population minimum density (γ). Notably, the best results are achieved when γ is maintained at an absolute value of 0.5, rather than scaling it with the vertex count $|\mathcal{V}|$. This suggests that a fixed minimum density is sufficient to ensure effective graph sparsity and clustering, even as the model scales, while also preventing over-sparsification (sparsity $\rightarrow 1.0$). In contrast, the population maximum density β benefits from scaling, with $\beta = 1/6 \cdot |\mathcal{V}|$ performing well across the Mu, Lt, and Ti scales. This configuration yields an average graph sparsity of approximately 30% to 60%, striking a balance between maintaining structural integrity and enabling efficient computation. These findings reinforce

the generalizability of the population regularization framework across scales while providing practical guidance for selecting γ and β values.

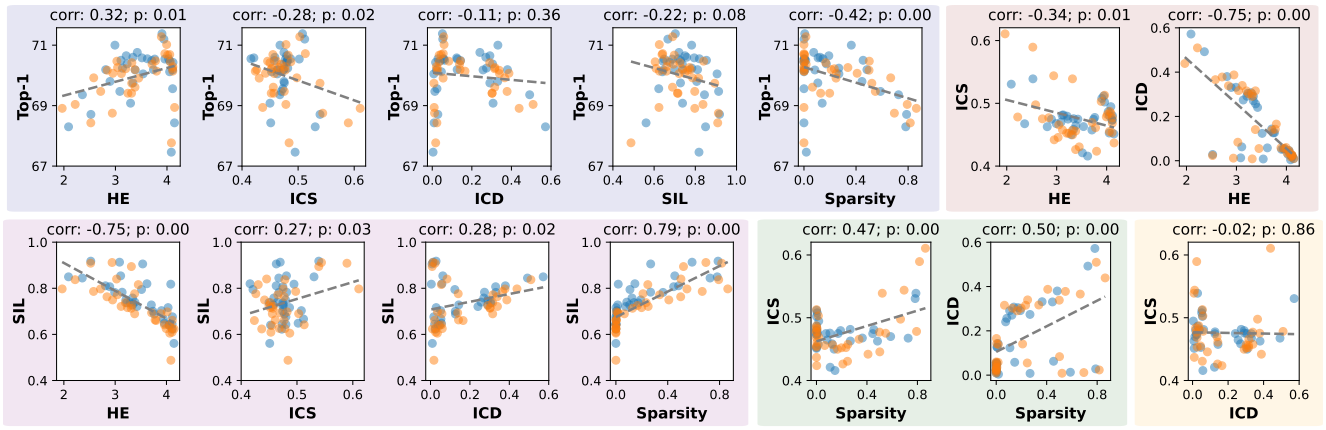
H.2. Correlation Analysis of Metrics

To further investigate the interactions between different metrics, we compute correlations across the HgVT-Mu population regularization sweep for both the standard Hadamard and modified Hadamard modulation methods. These correlations are visualized in Fig. 21, with graph quality metrics (HE, ICS, ICD, SIL, sparsity) analyzed in Fig. 21a and retrieval performance metrics (mAP@10 and 1NN-hit@10 for PS, VS, APS, AVS) in Fig. 21b. Each plot includes a best-fit trendline alongside the correlation coefficient and p-value to assess statistical significance.

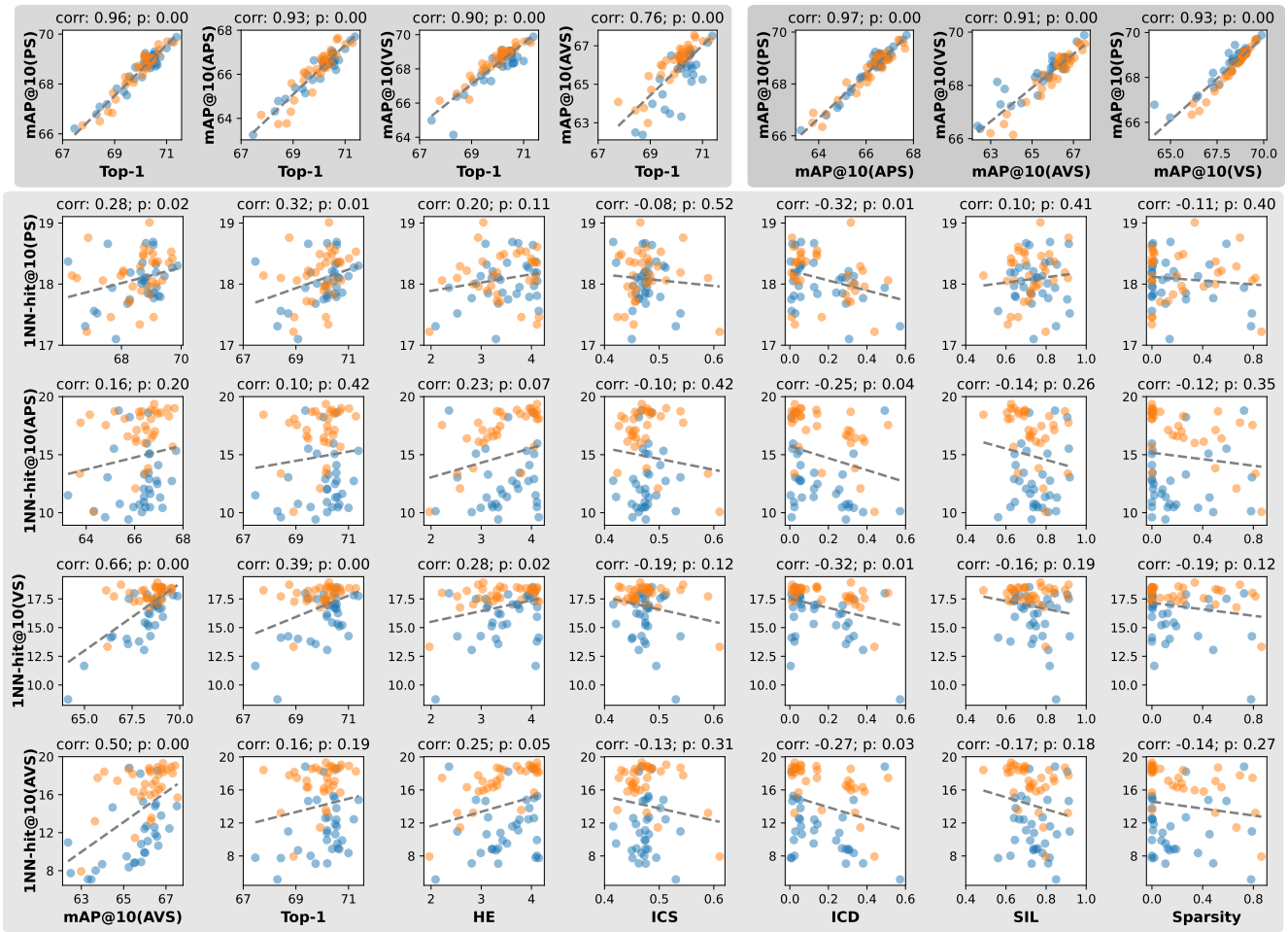
Graph Quality Metrics: Top-1 accuracy shows weak correlations with all graph quality metrics, positively with hyperedge entropy (HE) and negatively with all others, including sparsity. This supports the observation that maximally connected graphs tend to yield better Top-1 performance. SIL is negatively correlated with HE and positively correlated with sparsity, suggesting a trade-off between hyperedge feature variance and graph separation. Similarly, ICD is negatively correlated with HE, while ICS and ICD exhibit no correlation with each other. Most other interactions between graph quality metrics are relatively weak.

Retrieval Metrics: All mAP@10 metrics are highly correlated with Top-1 accuracy, with AVS exhibiting the largest variance. Adaptive methods (APS, AVS) are strongly correlated with their non-adaptive counterparts (PS, VS), while PS and VS also display strong mutual correlation. These relationships highlight consistent dependencies between retrieval metrics and Top-1 accuracy.

1NN-hit@10 Metrics: Acting as a proxy for semantic alignment with CLIP, the 1NN-hit@10 results reveal distinct groupings based on modulation type, with the modified Hadamard method outperforming the standard method. Interestingly, correlations in this category are generally weak, with the strongest observed between mAP@10 for the AVS method and 1NN-hit@10. This correlation is particularly notable when comparing VS and AVS within the 1NN-hit@10 metric. These findings suggest that while retrieval metrics align well with accuracy, their connection to semantic alignment is more nuanced and varies across methods.



(a) Graph Structure Correlations.

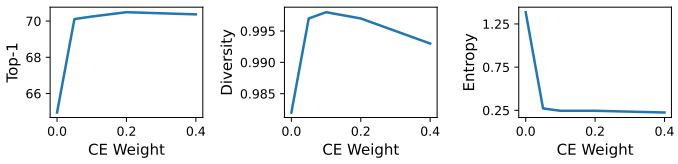


(b) Retrieval Accuracy Correlations.

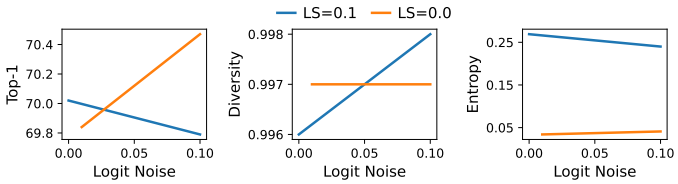
Figure 21. Comparing structural correlations obtained by the population regularization sweep on CIFAR-100 from Fig. 19. (a) measuring top-1 accuracy and intra-structural correlations; (b) showing structural correlations with image retrieval accuracy. Plotting both standard Hadamard (blue) and modified Hadamard (orange) soft membership modulation. Correlation coefficients and significance p-values plotted above each subplot, with correlation trendlines shown as gray-dashed lines.

H.3. Expert Pooling Regularization

Expert pooling regularization is evaluated on the HgVT-Mu model trained on CIFAR-100, focusing on the cross-entropy (CE) weight and logit noise injection strength. Fig. 23 examines these parameters, presenting Top-1 accuracy, expert diversity (where 1.0 indicates uniform expert utilization), and expert entropy (lower values indicate higher confidence). A CE weight of 0.1 achieves a good balance, yielding confident routing and high accuracy. Without the CE weight, expert entropy increases significantly, reflecting low-confidence routing that hinders performance. For logit noise injection, higher noise levels (10^{-1}) outperform label smoothing, improving both accuracy and diversity. This indicates that noise injection is a more effective regularization strategy, avoiding the higher entropy associated with label smoothing.



(a) Impact of CE Weight.



(b) Impact of Noise Injection.

Figure 23. Parameter sweep of Expert Edge hyperperameters for HgVT-Mu trained on CIFAR-100. (a) Varying cross-entropy loss weight; (b) varying logit noise injection strength with (LS=0.1) and without (LS=0.0) label smoothing. For both figures, showing top-1 prediction accuracy, diversity (1.0 indicates equal distribution among experts), and selection entropy (lower indicates higher confidence).

Table 13. Ablating Expert Edge pooling regularization methods for the HgVT-Mu model trained on CIFAR-100.

Density Loss	Label Smoothing	Dropout	Top-1	Diversity	Entropy
✓	✓	✓	70.25	0.998	0.245
✓	✓	✗	70.18	0.995	0.256
✓	✗	✓	69.81	0.987	0.039
✓	✗	✗	69.95	0.994	0.043
✗	✓	✓	66.37	0.0	1.386
✗	✓	✗	63.34	0.329	1.386
✗	✗	✓	64.54	0.323	1.386

Tab. 13 explores the combinatorial effects of diversity loss, label smoothing, and dropout regularization. Diversity loss proves essential, preventing expert collapse and achieving the highest diversity metric. Label smoothing and dropout individually have minor effects but, when combined, produce the best Top-1 accuracy and diversity results. However, label smoothing increases entropy, potentially reducing confidence. This is mitigated by omitting label smoothing and using higher logit noise instead, which preserves confidence while improving diversity and accuracy.

H.4. Repeated Blocks

We investigate the impact of repeating the final blocks in HgVT using the HgVT-Lt model on ImageNet-100. Inspired by hierarchical feature extraction in pyramidal models and the observed increase in sparsity with layer depth, we hypothesize that most hypergraph processing occurs earlier in the network, with later blocks primarily refining the structure. From this insight, we test whether repeating the final block can reduce the overall parameter count without degrading performance. Tab. 14 summarizes the results, confirming that repeating a single block maintains performance, while reducing parameter count. Meanwhile, repeating twice results in a slight accuracy decrease, suggesting that hypergraph refinement begins at the penultimate block. Notably, the FLOP count remains unchanged in all cases.

Table 14. Effect of repeating the final block on HgVT-Lt with a constant effective depth.

L	Repeats	Params	FLOPs	Top-1
12	0	6.62M	0.88G	82.23
11	1	6.23M	0.88G	82.65
10	2	5.74M	0.88G	81.92

H.5. Additional HgVT-Lt Model Ablations

Additional ablations on the HgVT-Lt model trained on ImageNet-100 explore various structural configurations. Tab. 15 lists these configurations, reporting their Top-1 accuracy, parameter count, and FLOPs. Fig. 24 visualizes the results, plotting accuracy against FLOPs, with marker size representing model size. The Pareto frontier is highlighted, alongside comparisons with ViG and ViHGNN, providing a reference point for FLOPs and parameter count.

The findings indicate that using split adjacency and feature matrices ($X_{\text{adj}} \neq X$) improves performance. Allocating more dimensions to the feature matrix than the adjacency matrix ($d_f > d_a$) strikes a balance between accuracy and computational overhead. Using more attention heads with smaller key dimensions ($d_k = 32$) outperforms fewer heads with a larger dimension. Furthermore, sharing the same feed-forward network (FFN) between edges and vertices reduces parameters with minimal

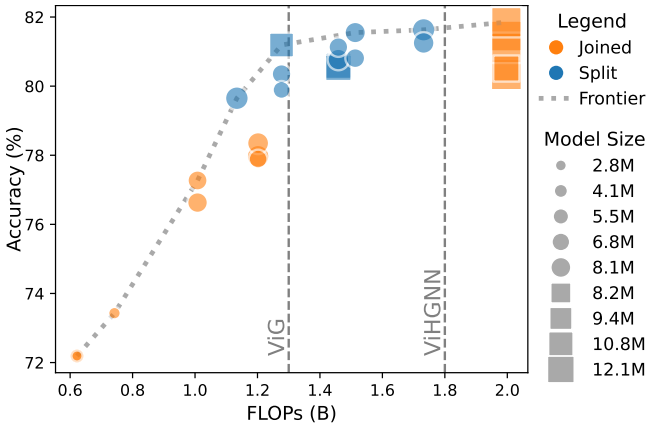


Figure 24. Showing ImageNet-100 classification accuracy vs forward compute (in FLOPs) for an architectural sweep of the HgVT-Lt model using expert pooling. Parameter count is shown by marker size, where models larger than ViHGNN-Ti [17] are represented by squares rather than circles. All FLOPs and Parameters are measured using the equivalent HgVT-Ti models on ImageNet-1k with expert pooling. Further showing models with joined ($\mathbf{X}_{\text{adj}}^{(*)} = \mathbf{X}^{(*)}$; orange), and split ($\mathbf{X}_{\text{adj}}^{(*)} \neq \mathbf{X}^{(*)}$; blue) adjacency features, along with the Pareto frontier.

Table 15. Architectural Ablations for HgVT-Lt trained on ImageNet-100. All experiments presented use average edge pooling.

$X_{\text{adj}} = X$	Joint FFN	L	d_f	d_a	h	d_k	Top-1	Params	FLOPs
✗	✗	10	96	96	3	32	81.19	8.3M	1.3G
✗	✗	10	128	64	4	32	80.59	9.2M	1.5G
✗	✗	10	64	128	2	32	80.59	7.7M	1.1G
✗	✗	10	128	64	2	64	80.43	9.2M	1.5G
✗	✗	10	64	128	1	64	80.25	7.7M	1.1G
✗	✓	10	128	64	4	32	80.77	6.6M	1.5G
✗	✓	10	96	96	3	32	80.35	5.7M	1.3G
✗	✓	12	128	64	4	32	81.63	7.6M	1.7G
✗	✓	12	96	96	3	32	81.55	6.5M	1.5G
✓	✗	10	96	96	3	32	72.19	4.0M	0.6G
✓	✗	10	128	128	4	32	77.27	5.9M	1.0G
✓	✗	12	128	128	4	32	78.35	6.8M	1.2G
✓	✓	12	128	128	4	32	77.89	5.4M	1.2G
✓	✗	10	192	96	6	32	81.85	11.8M	2.0G
✓	✗	10	192	96	3	64	81.11	11.8M	2.0G
✓	✓	10	192	96	3	64	80.51	9.1M	2.0G

accuracy loss. Several alternative configurations to the one chosen for HgVT-Lt are noted, offering trade-offs between computational overhead and accuracy for future scaling considerations.

I. Implementation Details

All models were trained using PyTorch with automatic mixed precision, leveraging the PyTorch-Lightning framework. Vertex self-attention was implemented efficiently using the xformers library [32], while edge attention utilized einsum operations reodered for memory efficiency with `torch.compile`. The Timm library [58] was employed for data augmentation, learning rate scheduling, and optimizer initialization, with the Fused AdamW optimizer from the Apex library [37].

Retrieval methods were implemented by storing precomputed features in HDF5 tables and conducting similarity searches directly on the GPU via PyTorch. The pooled embeddings of the full database were compact enough to reside in VRAM, enabling batch comparisons and efficient similarity sorting. Reranking computations were performed using Numpy on the shortlist features, eliminating the need to store these features on the GPU and maintaining computational efficiency.

I.1. Training Hyperparameters

Table 16. Details of data augmentation parameters, common to all runs.

Parameter	**Value**
Random Erase Mode	Pixel
Random Erase Probability	0.25
Random Erase Count	1
Label Smoothing	0.1
Mixup α	0.8
CutMix α	1.0
Mixup Probability	0.8
Mixup Switch probability	0.5
Mixup Mode	Batch
Repeat Augmentation Count	2
Color Jitter	0.4
Interpolation Mode	Random
Random Scale Range	[0.08, 1.0]
Random Aspect Ratio Range	[0.75, 1.33]
Random HFlip Probability	0.5
Auto-Agumentation Config.	`rand-m9-mstd0.5-inc1`

Table 17. Details of training hyper-parameters.

Parameter \ Scale →	**Mu**	**Lt**	**Mi**	**Ti**	**S**
Dataset	CIFAR100	ImageNet-100	ImageNet-1k	ImageNet-1k	ImageNet-1k
Resolution	32 x 32	160 x 160	224 x 224	224 x 224	224 x 224
Parameters	2.90M	6.82M	5.83M	7.76M	22.94M
Fwd. FLOPS	0.15G	0.92G	1.39G	1.80G	5.48G
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Peak Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3
Betas	[0.9, 0.999]	[0.9, 0.999]	[0.9, 0.999]	[0.9, 0.999]	[0.9, 0.999]
Eps	1e-8	1e-8	1e-8	1e-8	1e-8
Weight Decay	5e-2	5e-2	5e-2	5e-2	5e-2
Gradient Clip	1.0	1.0	1.0	1.0	1.0
Training Epochs	400	200	300	300	300
Warmup Epochs	10	16	10	10	10
Global Batch Size	512	512	1024	1024	1024
Grad. Accum. Steps	1	1	1	1	2
Training Hardware	1x A6000	1x A6000	2x A6000	2x A6000	2x A6000
Precision	bfloat16	bfloat16	bfloat16	bfloat16	bfloat16
Attn. Precision	float32	float32	float32	float32	float32
Training Time	2 Hours	8 Hours	122 Hours	139 Hours	255 Hours
Depth (L)	10	12	8	12	14
Feature Dim (d_f)	64	128	128	128	224
Adj. Dim (d_a)	64	64	64	64	96
Heads (h)	2	4	4	4	7
Joint FFN	True	True	True	True	True
$\mathbf{X}_{\text{adj}} = \mathbf{X}$	False	False	False	False	False
Patch Size	4	16	16	16	16
Image Verts. ($\|i\mathcal{V}\|$)	64	100	196	196	196
Virtual Verts. ($\|v\mathcal{V}\|$)	5	12	16	16	16
Primary Edges ($\|p\mathcal{E}\|$)	8	32	50	50	50
Virtual Edges ($\|v\mathcal{E}\|$)	4	6	8	8	8
Use Conv. Stem	True	True	True	True	True
Stochastic Path Drop	0.1	0.1	0.1	0.1	0.1
Class Dropout	0.1	0.0	0.0	0.0	0.0
Drop Decay	False	True	True	True	True
Pop Max (β)	10.05	20.7	36.04	36.04	36.04
Pop Min (γ)	0.5	0.5	0.5	0.5	0.5
λ_{POP}	1.0	1.0	1.0	1.0	1.0
λ_{DIV}	1.0	1.0	1.0	1.0	1.0
λ_{EXP}	1.0	1.0	1.0	1.0	1.0
Pooling Method	Expert	Expert+Image	Expert+Image	Expert+Image	Expert+Image
Expert Top-k	1	1	1	1	1
Expert λ_{CE}	0.1	0.1	0.1	0.1	0.1
Expert Noise	0.1	0.1	0.1	0.1	0.1
Expert Dropout	0.1	0.1	0.1	0.1	0.1
Expert Label Smoothing	0.0	0.0	0.0	0.0	0.0

J. Macro-Class Clustering with Expert Edge Pooling

This section provides taxonomy trees illustrating the macro-class clusters formed by our proposed expert pooling method. These clusters emerge as experts learn to select subsets of the hypergraph, revealing groupings aligned with high-level semantic categories.To illustrate, we present clusters from two models: HgVT-Lt, trained on ImageNet-100, and HgVT-S trained on ImageNet-1k. Given the reduced class count in ImageNet-100, the clusters for HgVT-Lt are more directly analyzable, whereas the larger taxonomy of ImageNet-1k consists of a broader set of categories.

Class-to-expert assignments are determined by histograms aggregated over the respective validation sets and follow a $2/3$ probability density rule: each class is assigned initially to its highest-probability expert, and subsequent experts are added if the most recently added expert contains less than $2/3$ of the remaining probability, until the total cumulative probability reaches 80%. For example, probability ranking $[54\%, 28\%, 12\%, 6\%]$ would assign the first two experts, while $[46\%, 24\%, 22\%, 8\%]$ would assign the first three experts. This allocation method produces a pattern of mostly single-expert assignments, tapering off with smaller groups assigned to two or more experts, which we visualize in the taxonomy trees in the following subsections.

J.1. HgVT-Lt on ImageNet-100

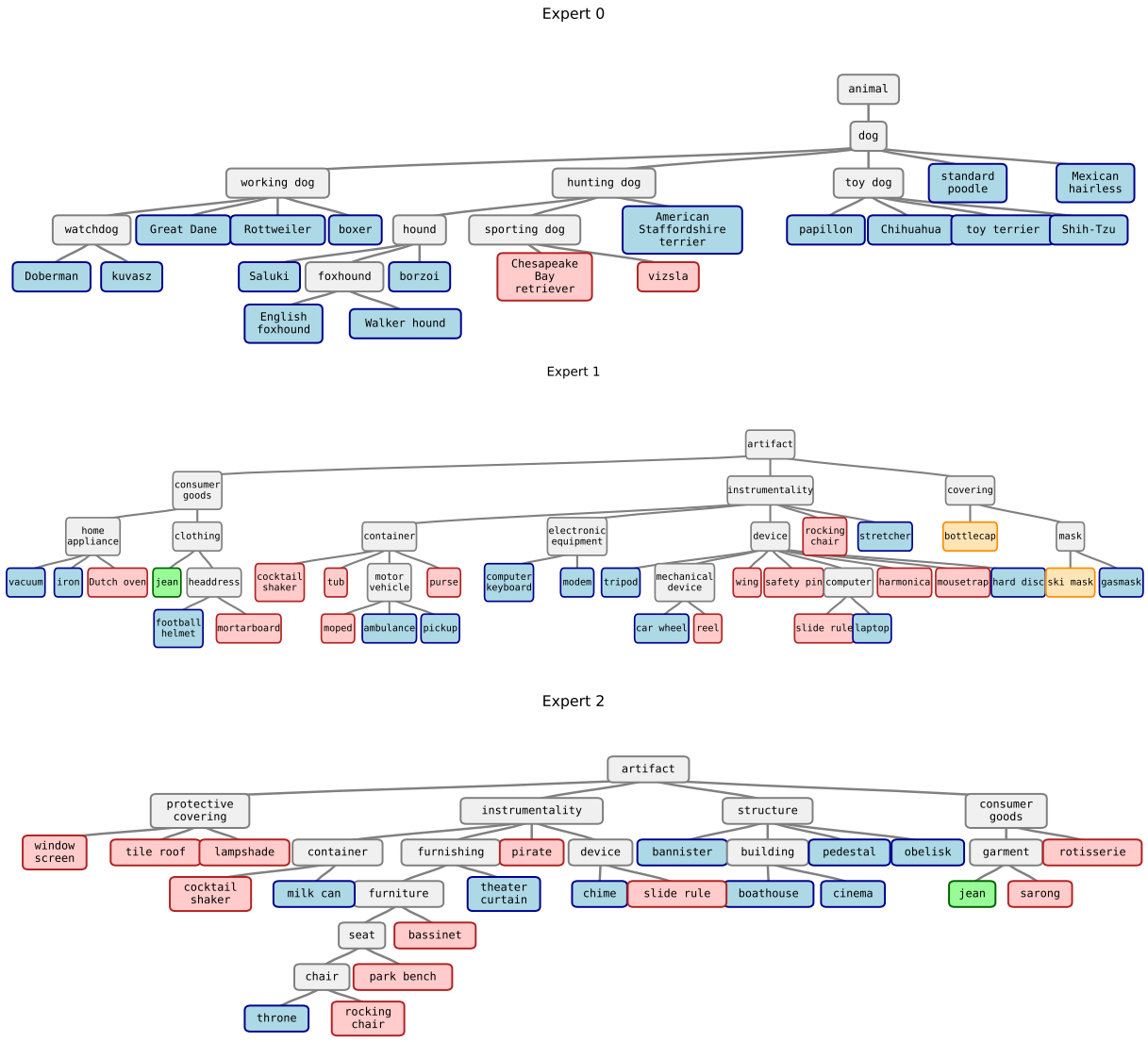


Figure 25. Macro-class clustering for the first three expert edges of HgVT-Lt on the ImageNet-100 validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges, (green) split over four edges.

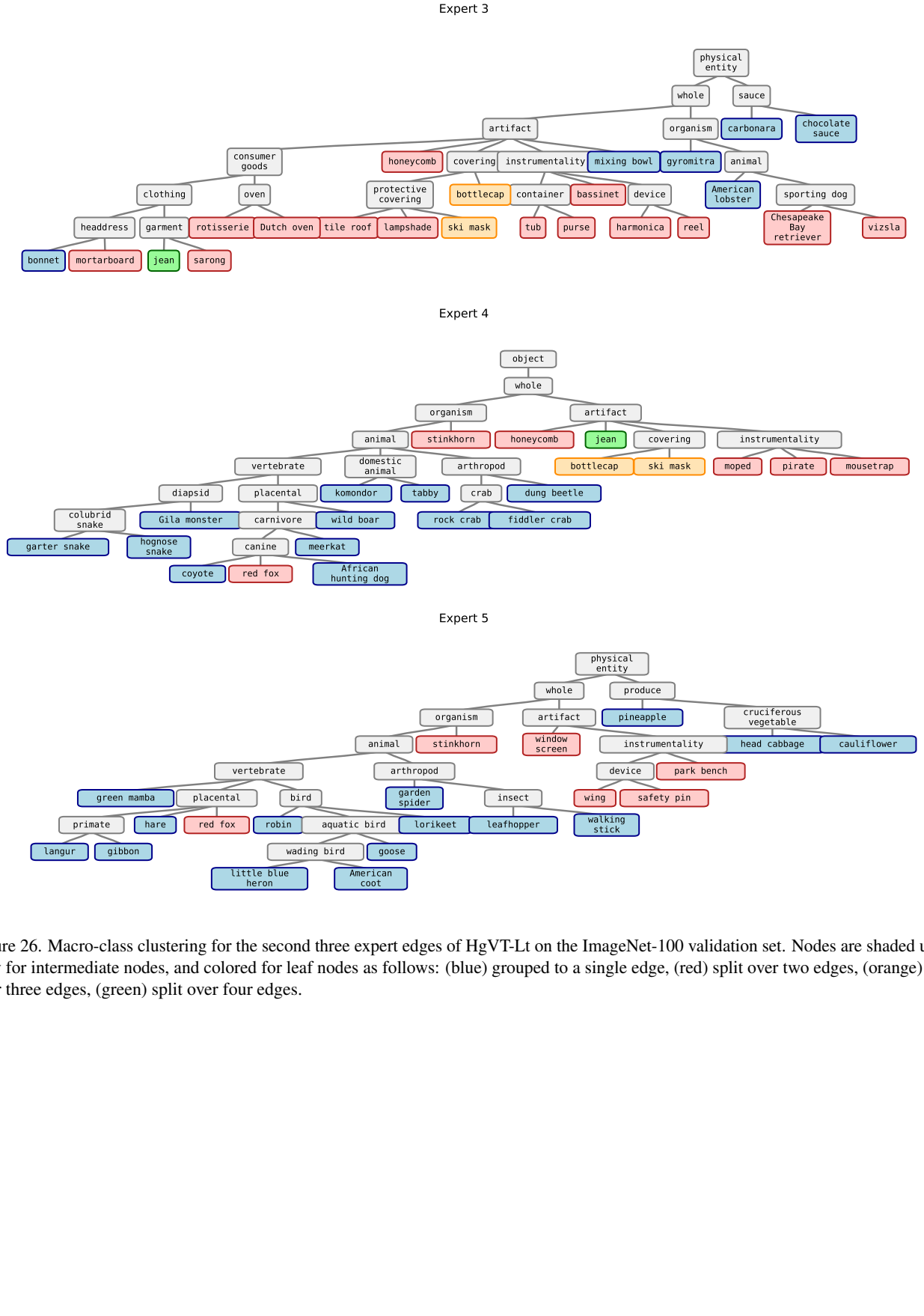


Figure 26. Macro-class clustering for the second three expert edges of HgVT-Lt on the ImageNet-100 validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges, (green) split over four edges.

J.2. HgVT-S on ImageNet-1k

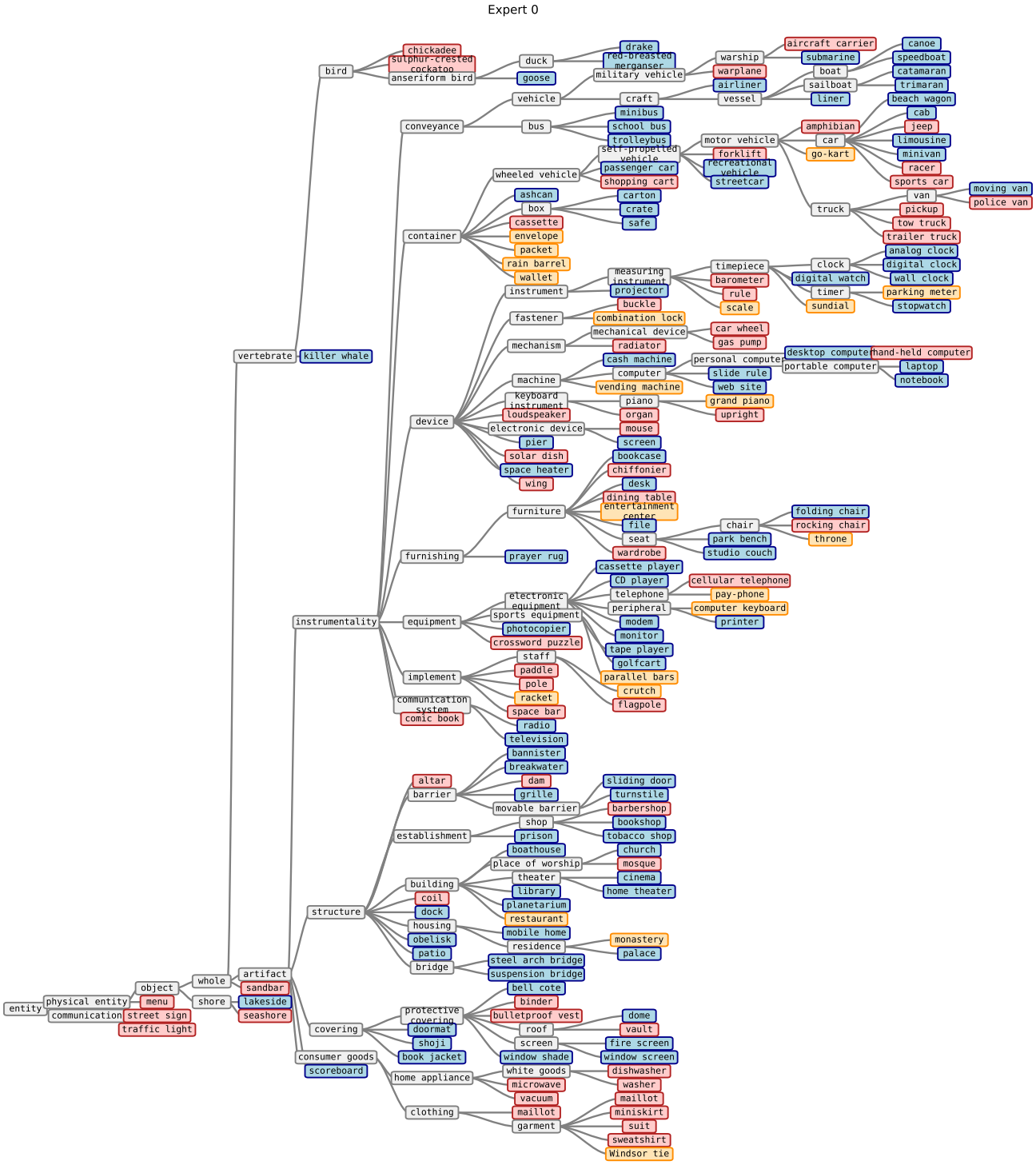


Figure 27. Macro-class clustering for expert $0/8$ of HgVT-S on the ImageNet-1k validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges.

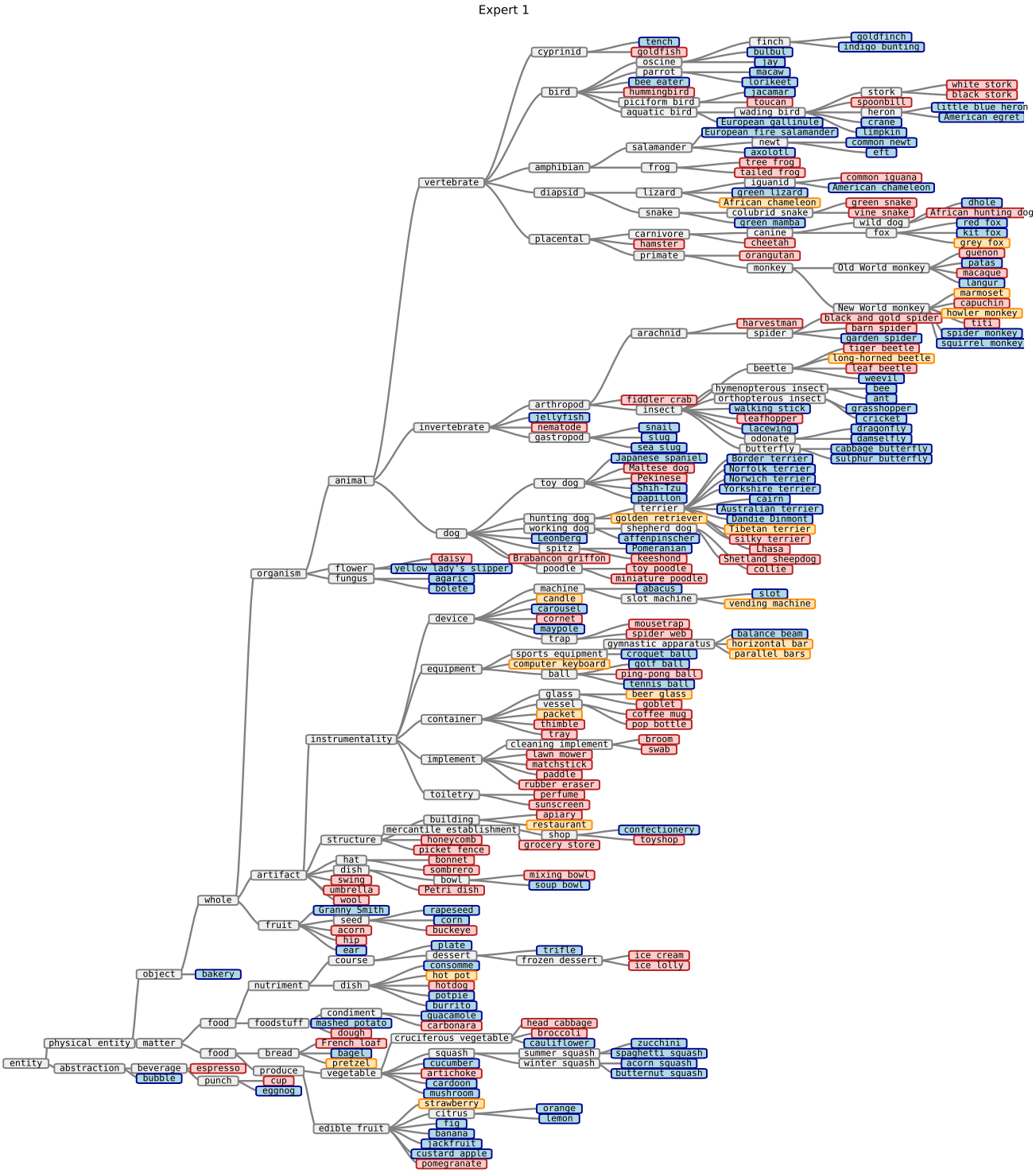


Figure 28. Macro-class clustering for expert $1/8$ of HgVT-S on the ImageNet-1k validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges.

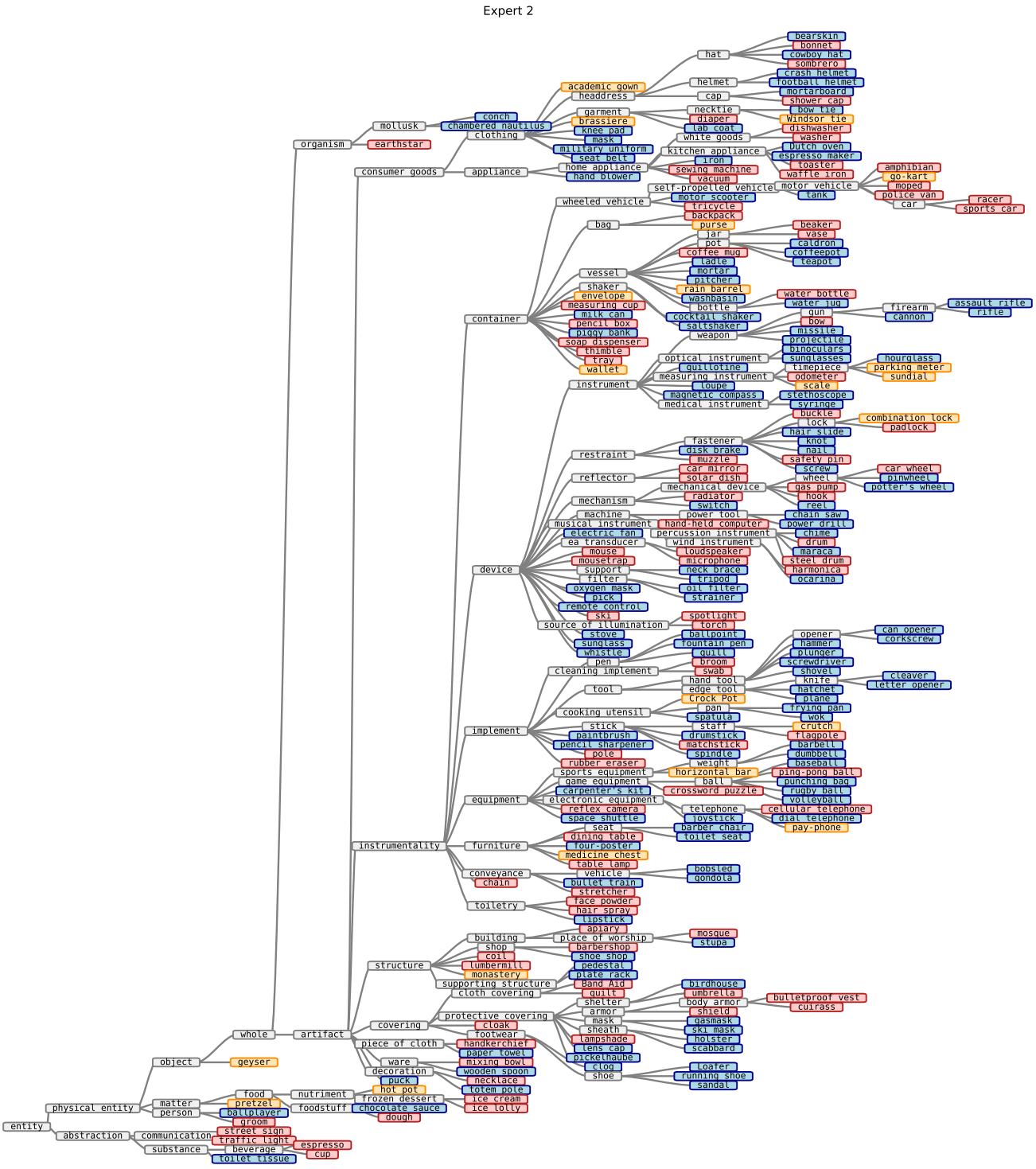


Figure 29. Macro-class clustering for expert $2/8$ of HgVT-S on the ImageNet-1k validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges.

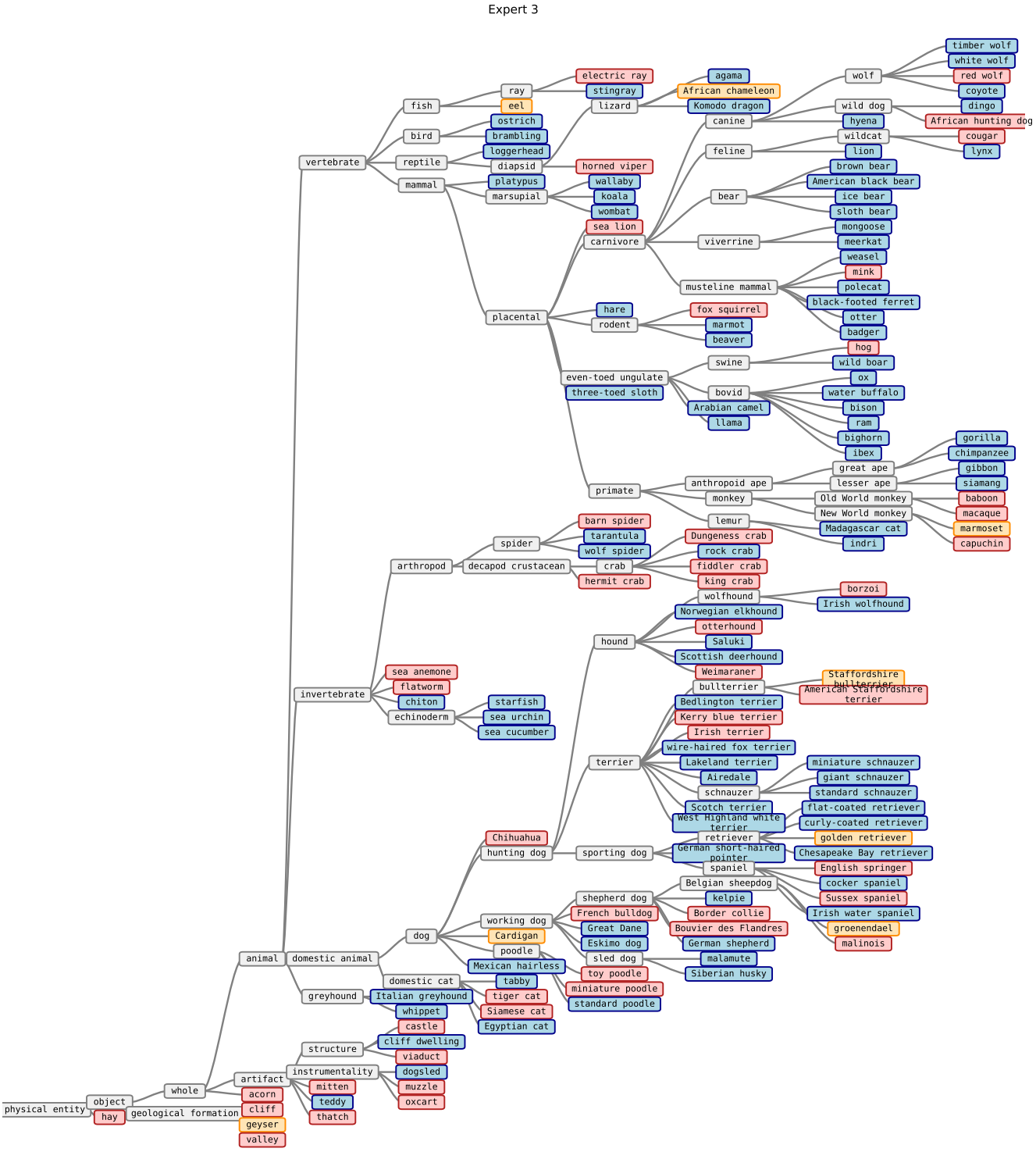


Figure 30. Macro-class clustering for expert 3/8 of HgVT-S on the ImageNet-1k validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges.

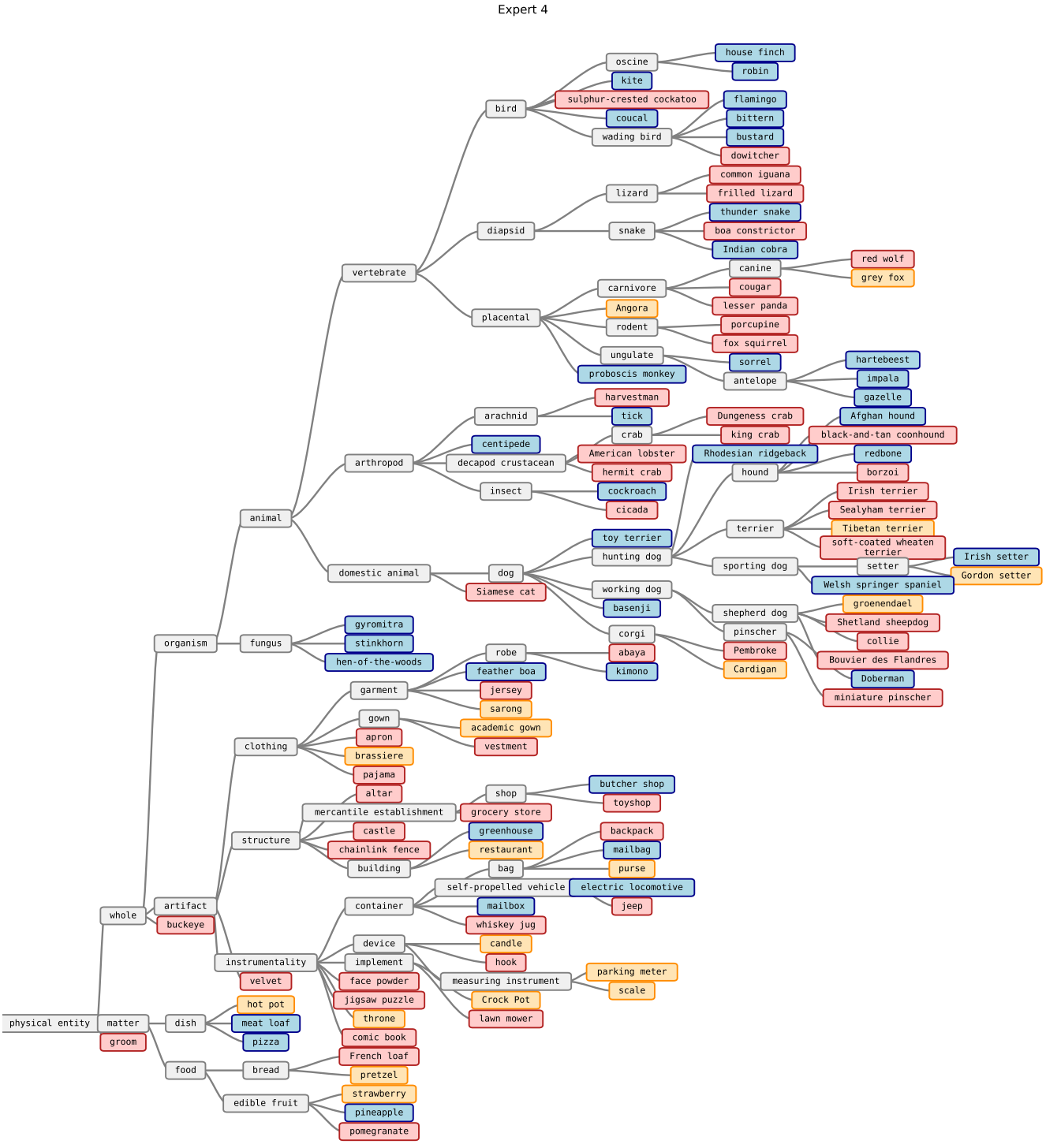


Figure 31. Macro-class clustering for expert $4/8$ of HgVT-S on the ImageNet-1k validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges.

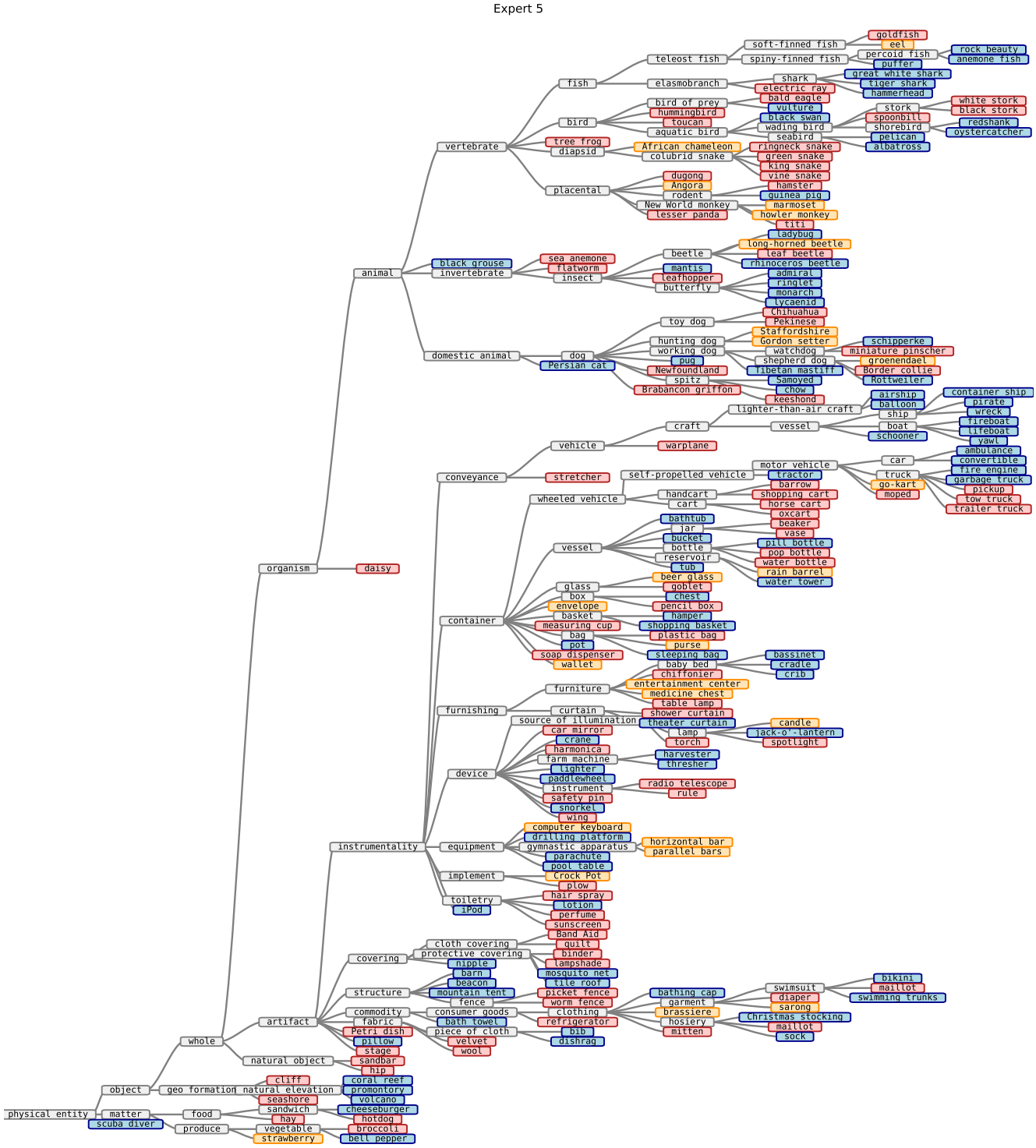


Figure 32. Macro-class clustering for expert $5/8$ of HgVT-S on the ImageNet-1k validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges.

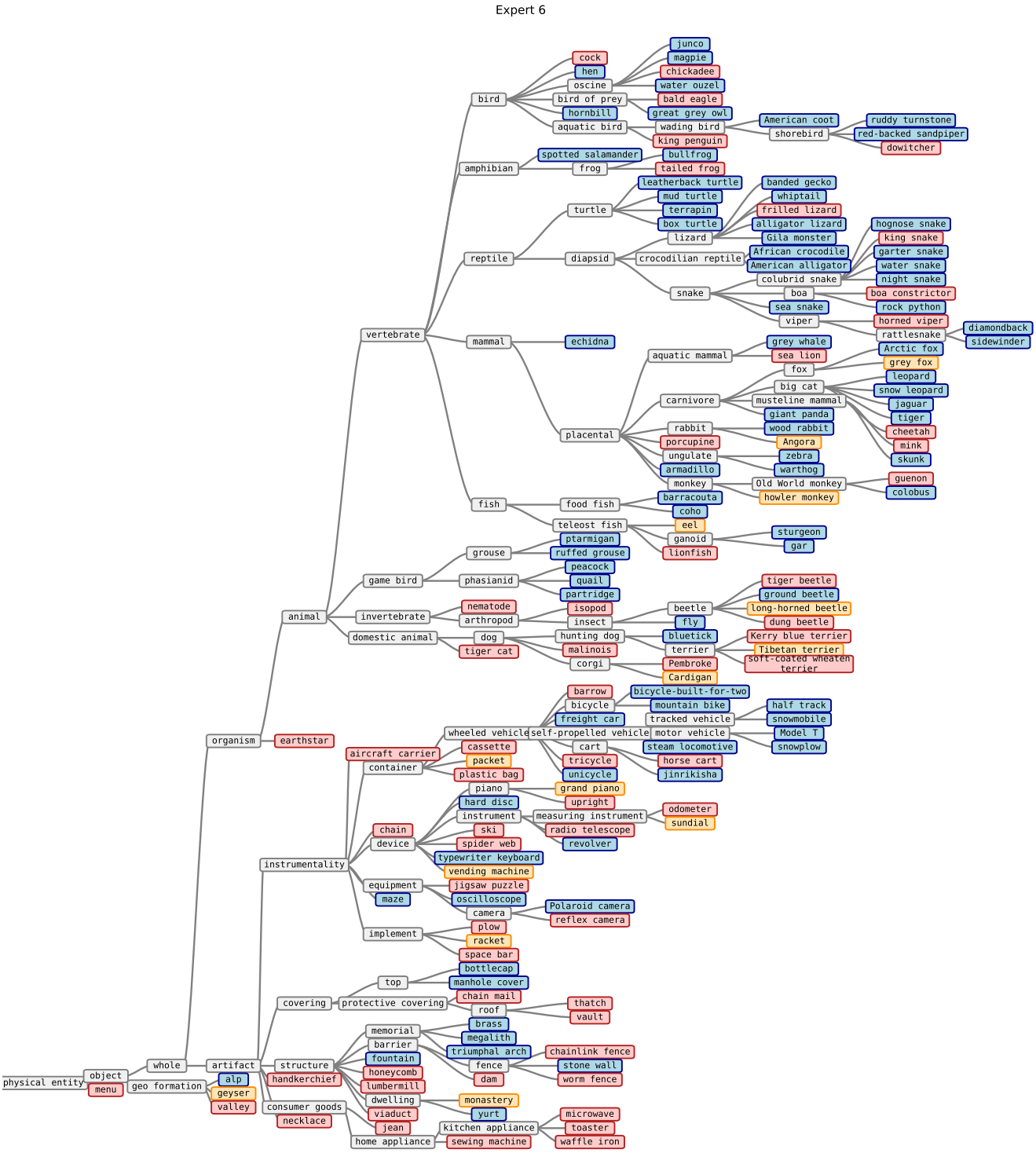


Figure 33. Macro-class clustering for expert $6/8$ of HgVT-S on the ImageNet-1k validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges.

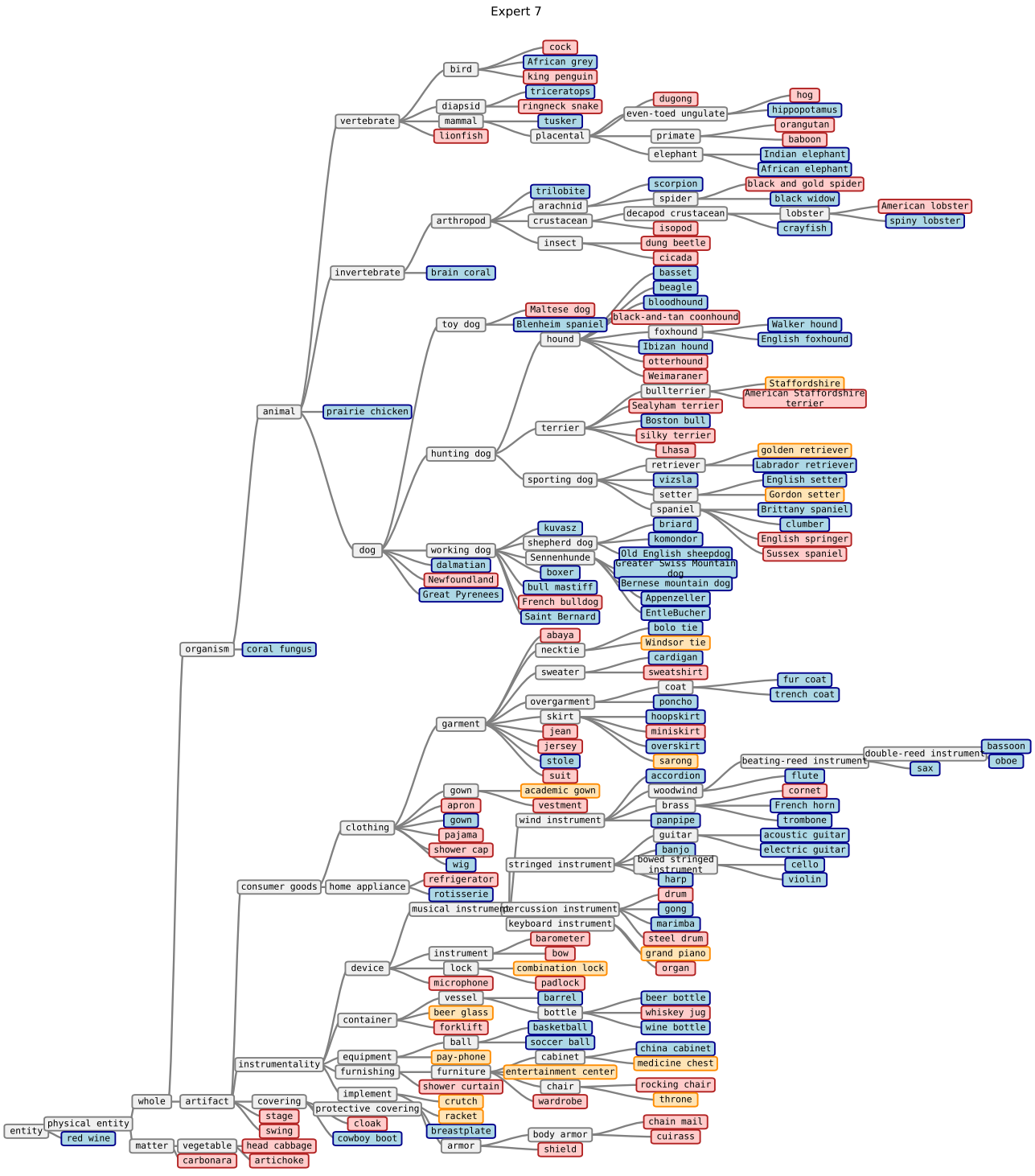


Figure 34. Macro-class clustering for expert 7/8 of HgVT-S on the ImageNet-1k validation set. Nodes are shaded using gray for intermediate nodes, and colored for leaf nodes as follows: (blue) grouped to a single edge, (red) split over two edges, (orange) split over three edges.