Learning Eigenstructures of Unstructured Data Manifolds

Supplementary Material

A. Additional Theory

A.1. PCA Alternative Formulations

Without loss of generality, assume $\mathbb{E}(f) = 0$, as is the case for our signals uniformly distributed in $\mathcal{C}_L = \{f; \|f\|_L \leq 1\}$. Denote $C = \mathbb{E}(ff^\top)$ as the covariance matrix of distributed functions f. In our case for Theorem 3.2, $C = R_L$ as $f \sim \mathcal{U}(\mathcal{S})$. The classical iterative optimization formulation for PCA is the following. PCA finds the orthonormal basis b that maximizes iteratively over k the Rayleigh quotient

$$\max_{b_k \text{ s.t. } \begin{cases} \|b_k\|_2=1 \\ b_k \perp \{b_1,\dots,b_{k-1}\} \end{cases}} b_k^\top C b_k. \tag{2}$$

The solution to this Rayleigh optimization, and thus to PCA, is given by the eigenvectors of C in decreasing order of eigenvalues.

Another alternative formulation exists, where the PCA optimization objective is given as the iterative minimization of the expected squared approximation error, or variance, in a basis b

$$\min_{b_k \text{ s.t. } \begin{cases} \|b_k\|_2=1 \\ b_k \perp \{b_1,\dots,b_{k-1}\} \end{cases}} \mathbb{E}\left(\left\| f - \sum_{i=1}^{k} \langle f, b_i \rangle b_i \right\|^2\right). \tag{3}$$

Indeed, let us remind the classical link between the two formulations. Denoting $B_k \in \mathbb{R}^{d \times k}$ with columns $b_1, \dots, b_k$, with $B_k^\top B_k = I_k$, the expected squared approximation error can be rewritten as

$$\mathbb{E}\left(\left\| f - \sum_{i=1}^{k} \langle f, b_i \rangle b_i \right\|^2\right) = \mathbb{E}(\|(I - B_k B_k^\top) f\|^2) \tag{4}$$

$$= \operatorname{tr}(\mathbb{E}(f^\top (I - B_k B_k^\top)^\top (I - B_k B_k^\top) f)) \tag{5}$$

$$= \operatorname{tr}(\mathbb{E}(f^\top (I - B_k B_k^\top) f)) \tag{6}$$

$$= \operatorname{tr}(\mathbb{E}((I - B_k B_k^\top) f f^\top)) \tag{7}$$

$$= \operatorname{tr}((I - B_k B_k^\top) C) \tag{8}$$

$$= \operatorname{tr}(C) - \operatorname{tr}(B_k B_k^\top C) \tag{9}$$

$$= \operatorname{tr}(C) - \operatorname{tr}(B_k^\top C B_k) \tag{10}$$

$$= \operatorname{tr}(C) - \sum_{i=1}^{k} b_i^\top C b_i \tag{11}$$

As C is a constant, minimizing the expected squared approximation error is thus the same as maximizing the cumulative sum of Rayleigh quotients of C. Working iteratively over k, we get that the optimal solution b for both formulations is given by the eigenvectors of C in decreasing eigenvalue order. □

A.2. Normalized Operators and Eigenstructures

We here generalize the relationship between the normalized symmetric Laplacian compared to the unnormalized one regarding its first eigenvector and how it provides the metric.

For any operator A, we can define an analogous normalized operator $A_{\text{norm}} = NAN^{-1}$, where N is a matrix used for normalization. For instance $N = M^{\frac{1}{2}}$ for the normalized Laplacian L_{norm} and $A = L$. Both operators A and A_{norm} share the same eigenvalues but different eigenvectors, related by the normalization matrix N. Indeed, if λ_i, e_i^{norm} are eigenvalues and eigenvectors of A_{norm}, then $AN^{-1}e_i = N^{-1}NAN^{-1}e_i = N^{-1}A_{\text{norm}}e_i = \lambda_i N^{-1}e_i$, meaning that $e_i = N^{-1}e_i^{\text{norm}}$ are the eigenvectors of the unnormalized operator A. As such, choosing $N = M^{\frac{1}{2}}$ to be given by the metric as for the normalized Laplacian, the eigenvectors of the normalized Laplacian are related to the unnormalized one by the metric $e_i^{\text{norm}} = M^{\frac{1}{2}}e_i$. Note that A_{norm} is symmetric if $A = N^{-2}\Sigma$ for symmetric Σ and N, which is the case for the Laplacians where $\Sigma = S$ and $N = M^{\frac{1}{2}}$ is diagonal, as then $A_{\text{norm}} = N^{-1}\Sigma N^{-1}$, and so by the spectral theorem e_i^{norm} is Euclidean-orthogonal whereas e_i is $N^\top N$-orthogonal.

Of note, as for the Laplacians, if $e_1 = \mathbf{1}$ is in the nullspace of the unnormalized operator A, then $e_1^{\text{norm}} = N\mathbf{1}$ is in the nullspace of A_{norm}. Remarkably, if N is a diagonal matrix (like $M^{\frac{1}{2}}$), if the nullspace is of dimension one, and if we learn to find the nullspace by learning the eigenbasis directly, then we can compute N directly from it as $\operatorname{diag}(N) = e_i^{\text{norm}}$. For most operators of interest, the assumptions on A hold (for connected relationship graphs). This is because operators of interest are usually linear and differential, i.e. linear combinations with constant coefficients of the derivative operator $A = \sum_{|k| \leq m} \alpha_k \partial^k$, usually without a 0-th order term, and constant functions are zeroed out by derivatives.

As such, our methodology learning a Euclidean orthogonal basis directly is well-suited to capture eigenstructures of implicit operators with Euclidean orthogonal eigenvectors and a single nullspace eigenvector dimension given by a diagonally scaled version of the $\mathbf{1}$ vector. The normalized Laplacian falls into this category, yet it is just one out of the many other interesting operators satisfying these assumptions.

B. Additional Details on Our Framework

B.1. Projecting onto Optimal Reconstruction Bases

To approximate probe signals, we need to compute projections onto $\mathbf{Q}_k$. However, Euclidean, that is uniform, orthogonal projection will not account for the non-uniform geometry of the sampled manifold due to non-uniform sampling density. To overcome this issue, we resolve to the M-orthogonal projection onto the columns of $\mathbf{Q}_k$. As they are the output of a QR-decomposition, these vectors are Euclidean orthogonal and not M-orthogonal, yet, we want to M-orthogonally project onto their span. To do this, the M-orthogonally projected functions can be rewritten as

$$\mathbf{f}_{\text{proj},k} \quad = \quad \sum_{i=1}^{k} c_i^{(k)} \mathbf{q}_i \quad = \quad \mathbf{Q}_k \mathbf{c}^{(k)}, \tag{12}$$

where $\mathbf{c}^{(k)} = [c_1^{(k)}, \dots, c_k^{(k)}]^\top \in \mathbb{R}^k$ contains the projection coefficients. The coefficients $\mathbf{c}^{(k)}$ are determined by requiring that the residual $(\mathbf{f} - \mathbf{f}_{\text{proj},k})$ is M-orthogonal to all basis vectors

$$(\mathbf{f} - \mathbf{f}_{\text{proj},k})^\top M \mathbf{q}_i \; = \; 0 \quad \text{for} \; \; i \; = \; 1, \dots, k. \tag{13}$$

This orthogonality condition yields the system

$$\mathbf{q}_i^\top M \mathbf{f} \quad = \quad \sum_{j=1}^{k} c_j^{(k)} (\mathbf{q}_i^\top M \mathbf{q}_j). \tag{14}$$

In matrix form, this becomes

$$\mathbf{G}^{(k)} \mathbf{c}^{(k)} \quad = \quad \mathbf{b}^{(k)}, \tag{15}$$

where $\mathbf{G}^{(k)} \in \mathbb{R}^{k \times k}$ is the Gram matrix with entries $G_{ij}^{(k)} = \mathbf{q}_i^\top M \mathbf{q}_j$, and $\mathbf{b}^{(k)} \in \mathbb{R}^k$ is a vector with entries $b_i = \mathbf{q}_i^\top M \mathbf{f}$. Since $\mathbf{G}^{(k)} = \mathbf{Q}_k^\top M \mathbf{Q}_k$ is symmetric positive definite[2], we can efficiently solve this linear system using a differentiable solver to obtain the coefficients $\mathbf{c}^{(k)}$[3]. Crucially, $\mathbf{G}^{(k)} \in \mathbb{R}^k \times k$ is small, where $k \ll n$ is the number of eigenvectors used for projection, making the system computationally efficient and independent of the input size n. Note that $\mathbf{c}^{(k)}$ is not the truncation of $\mathbf{c}^{(K)}$ to its first k terms since $\mathbf{Q}_k$ is not orthogonal for the used M-weighted scalar product. These independent systems are batched and solved concurrently for all $k \leq K$.

B.2. Replacing the M-norm with the Euclidean norm for Approximation Errors

While the theoretical framework calls for the M-norm in the error computation (in both Theorems 3.1 and 3.2), we deliberately use the 2-norm in our loss (Eq. (1)). This hybrid approach, using M-weighted projection with 2-norm error, creates an unsupervised mechanism for learning a density-related mass matrix. Since the 2-norm weights all points equally regardless of sampling density, regions with different sampling densities contribute differently to the loss, implicitly encouraging the network to learn mass elements connected to the ambient space's Euclidean-induced volume (or area on surface manifolds) of the samples. We also experimented with using the M-norm for error computation as prescribed by theory. However, this resulted in unstable training, numerical issues (NaNs), and even collapses of the metric to 0 to naively minimise $\mathcal{L}_{\text{rec}}$. This confirms the necessity of our hybrid approach.

[2] As the columns of $\boldsymbol{Q}_k$ are linearly independent thanks to the QR-decomposition and M is positive definite.

[3] Avoiding the explicit inversion in the formula $\mathbf{c} = (\mathbf{G}^{(k)})^{-1}\mathbf{b}$.

C. Additional Experimental Details and Results

C.1. Toy 1D Segment Manifold

In this toy setting we work on a single point cloud. We uniformly grid sample $n = 100$ points $x_i = \frac{i}{n-1}$ for $i = 0, \dots, n-1$. Probe signals $\mathbf{f}_i$ are generated by drawing i.i.d. noise $\mathbf{g}_i \in \mathbb{R}^n$ and applying a few steps of 1D Gaussian smoothing on the kNN graph with 16 neighbors, yielding smooth probes $\mathbf{f}_i$. Our lightweight MLP consists in 3 hidden layers with width 64 and ReLU activations. We only compute the first $K = 5$ basis vectors. Wet train over batches of $m = 500000$ probe functions using the Adam [64] optimizer with an initial learning rate of $\eta = 10^{-2}$, a step scheduler with parameter $\gamma = 0.1$ at 30% and 70%.

C.2. 2D Surface and 3D Volume Manifolds

C.2.1. Datasets

Overfitting setting. We use shapes from [90] commonly used for benchmarking geometry processing works. We discard the mesh connectivity to work with point clouds.

Generalization setting. In order to train a foundation model for estimating spectral bases on 3D pointclouds, we unsupervisedly trained our model on many datasets covering a large spectrum of types of shapes: SURREAL [101] for synthetic human bodies, SHREC'21 Protein Domains [70] for biological structures, ABC [66] for CAD models, FAUST [16] for real human scans, and DeformingThings4D [75] for dynamic non-rigid surfaces. Evaluation is performed on well-established evaluation benchmarks covering many challenging settings: DefTransfer [122], MPZ14 [90], SHREC'07 Watertight [48], SHREC'14 Human Models [99], SHREC'15 Non-Rigid [76], SHREC'16 Topological Noise (TopKids) [69], SHREC'19 Humans [86], SHREC'20 Non-Isometric (SHREC'20-NI) [41], SHREC'20 Geometric Reliefs (SHREC'20-GR) [127], and Thingi10K [146]. All

shapes in the datasets, both in train and evaluation, originally come with mesh connectivity. When using our method, we remove all this oracle connectivity information to keep only unstructured point clouds. We also evaluate, but not train, on volumetric point clouds generated from surface meshes in SHREC'20 Non-Isometric [41]. To generate this data, we perform volumetric sampling using Kaolin's [4] [46] GPU-accelerated approach. Random candidate points are uniformly sampled within the mesh's axis-aligned bounding box (with 5% padding), then for each point we compute the unsigned distance to the nearest mesh surface and determine inside/outside classification via ray casting[5]. Points classified as interior are retained and subsampled to the target count, optionally combined with surface vertices to form the final volumetric point cloud.

C.2.2. Implementation – Generalization Setting

The following implementation details describe our generalization model that learns to predict a spectral basis resembling Laplacian eigenfunctions across diverse 3D shapes, rather than overfitting to a single geometry. The model is trained on a large-scale dataset combining multiple datasets to generalize to unseen shapes at test time.

Network Architecture. Our 3D model processes point clouds through a three-stage pipeline. First, raw 3D coordinates are passed through a preprocessing MLP with architecture [3 $\rightarrow$ 64 $\rightarrow$ 256 $\rightarrow$ 1024] using GELU activations [57] and layer normalization [8]. The resulting 1024-dimensional features are then processed by an 8-layer Transformer encoder with 32 attention heads, where each point attends to all other points in the point cloud via global self-attention. The Transformer uses $d_{\text{model}} = 1024$, feedforward dimension of 1024, GELU activations, and processes sequences in batch-first format. Finally, a lightweight output MLP [1024 $\rightarrow$ 256 $\rightarrow$ 50] with GELU activations and layer normalization predicts the 50 feature vectors for each point (prior to QR decomposition to get the estimated normalized spectral basis).

Training Configuration. We optimize with AdamW [82] (learning rate 5×10^{-4}) and cosine annealing schedule decaying to 1×10^{-5} over 100 epochs. We apply gradient clipping at 0.01 and accumulate gradients over 2 batches. The training set consists of approximately 200000 3D shapes (file sizes 0.2-7MB) from diverse datasets with a batch size of 30.

Data Processing. Each mesh is sampled to 1500 points using Farthest Point Sampling (FPS) [42, 49], with random initialization during training and fixed initialization during validation to ensure reproducibility. We generate 500 scalar fields per shape by sampling uniform random values in the range [-1, 1] at each vertex. Before computing the reconstruction loss, scalar fields are smoothed for 20 iterations using Gaussian kernel convolution with $\sigma = 0.1$ over a k-nearest neighbor graph with $k = 15$. The kNN graph construction uses Euclidean distance (not cosine similarity) and includes self-loops. Oracle cotangent Laplacian eigendecompositions are precomputed and cached to accelerate the validation phase, where we monitor cosine similarity between predicted and oracle eigenvectors.

Computational Resources. All experiments were conducted on 8 GPUs (type NVIDIA A100-SXM4-40GB) with automatic mixed precision [89] training enabled through PyTorch Lightning [43, 96].

C.2.3. Implementation – Overfitting setting

The following implementation details describe the configuration for overfitting the eigendecomposition of a specific 3D shape. This setup employs a smaller architecture to achieve maximum accuracy for a single target geometry.

Network Architecture. The architecture for single-shape overfitting uses a more compact design compared to the generalization model. Raw 3D coordinates are processed through a preprocessing MLP with architecture [3 $\rightarrow$ 32 $\rightarrow$ 128 $\rightarrow$ 256] using GELU activations and layer normalization. The core network consists of an 8-layer Transformer encoder with 8 attention heads (compared to 32 for generalization), $d_{\text{model}} = 256$, and feedforward dimension of 256. The output head is a simple two-layer MLP [256 $\rightarrow$ 50] with GELU activation and layer normalization that directly predicts the 50 Laplacian eigenvectors.

Training Configuration. We optimize with AdamW (learning rate 0.001) and cosine annealing schedule decaying to 1×10^{-5} over 5000 epochs, training for a total of 12,000 epochs. Gradient clipping is applied at 0.01. The training consists of a single mesh with batch size 1, allowing the model to specialize completely to the target geometry.

Data Processing. The single training mesh is sampled to 4000 points using Farthest Point Sampling (FPS) with random initialization at each iteration to provide variation during training. We use a higher point count compared to the generalization setting (4000 vs 1500) since batch size 1 requires significantly less GPU memory, allowing for denser point sampling. We generate 1500 scalar fields per shape by sampling uniform random values in the range [-1, 1] at each vertex. Before computing the reconstruction loss,

[4] https://github.com/NVIDIAGameWorks/kaolin

[5] Shooting a ray and counting mesh intersections, with odd counts indicating interior points.

each scalar field is smoothed for 40 iterations using Gaussian kernel convolution over a k-nearest neighbor graph with $k = 10$, where σ is randomly sampled from the range [0.01, 0.2] independently for each scalar field. We employ this range of smoothing scales rather than a fixed value because we empirically observed improved results across various shapes. We speculate that sampling different σ values allows the network to sense the shape's manifold at multiple scales, which helps generate a statistically sufficient number of smooth functions that are smooth with respect to the surface metric – a requirement for optimal approximation. Empirically, we found that using a range of σ values does not contribute significantly to the generalization case, suggesting this multi-scale sensing is particularly beneficial when overfitting to a specific geometry. The kNN graph uses Euclidean distance and does not include self-loops. Validation is performed every 200 epochs, with model checkpoints saved at the same frequency.

Computational Resources. Experiments were conducted on a single NVIDIA GPU using PyTorch Lightning. The compact architecture and small batch size make this configuration accessible for consumer-grade hardware such as an RTX 3090, requiring significantly fewer computational resources than the generalization model.

Eigenvalues at Inference Time. Our model is designed to compute a spectral basis. However, we can recover without additional cost as a byproduct the associated eigenvalues of the implicit optimal-reconstruction operator. It is given by the invert of the worse approximation error. Depending on how we generate probe functions at inference time, we get different errors and thus different eigenvalues. Our methodology to generate probe functions depends on three hyperparameters: the number of nearest neighbors k in the kNN graph, the number of smoothing iterations, and the smoothing scale σ. By default, we propose to take at inference time the fixed hyperparameters $k = 70$, 48 iterations, and $\sigma = 0.101$. These numbers gave us the minimal mean relative eigenvalue discrepancy with the oracle cotangent Laplacian's eigenvalues across all tested shapes. However, we also manually tuned these hyperparameters per shape and provide in Tab. 2 the best ones we found. Unless specified otherwise, presented results used the tuned version rather than the fixed one. In future work we intend to provide an automatic mechanism for finding the best hyperparameters per shape. For instance, as probe functions depend differentiably on σ, it could be learned via descent strategies.

C.2.4. Additional Results

Due to page-length constraints, we here provide additional results to those in the main paper.

Table 2. Optimal hyperparameters for generating probe functions in the single-shape overfitting setting when estimating eigenvalues. For each shape, we report the k-nearest neighbors (k), number of smoothing iterations, and Gaussian kernel bandwidth (σ) that yielded the best alignment with the oracle cotangent Laplacian's eigenvalues.

Shape	k	**Iterations**	σ
Armadillo	45	111	0.194
Beetle	70	120	0.120
Bimba	33	57	0.101
Botijo	27	93	0.120
David	33	48	0.138
Dente	21	75	0.101
Dragon	70	48	0.269
Elephant	57	75	0.157
Eros	27	102	0.138
Fertility	63	40	0.176
Heptoroid	39	40	0.213
Horse	70	75	0.194
Kitten	27	66	0.120
Knot	15	66	0.101
Laurent Hand	21	102	0.101
Lion	39	111	0.176
Master Cylinder	15	102	0.269
Pegaso	70	120	0.213
Teddy	45	48	0.138
Woodenfish	70	120	0.232
Wrench	70	120	0.082

Overfitting setting. In Figs. 11 to 14, we provide additional visualizations of the learned unnormalized spectral basis $\mathbf{v}_i$ and obtained spectral filtering, generalizing Fig. 3 to other shapes. We also plot on a few shapes in Figs. 15 to 19 all the first 50 spectral basis vectors (excluding the first constant one $\mathbf{v}_1$), i.e. $\mathbf{v}_2, \dots, \mathbf{v}_{50}$. In Fig. 21, we plot eigenvalue curves on other shapes than in Fig. 4 in the tuned hyperparameter setting for generating inference probe functions, along with those obtained in the fixed hyperparameter setting in Fig. 22. In Fig. 20, we display more estimated metrics M from $\mathbf{q}_1$ on many more shapes than in Fig. 5. In Tab. 3, we give further quantitative results between our estimated eigendecomposition and the oracle cotangent one[6] for the average cosine similarity and the eigenvalue discrepancy between both bases, extending Tab. 1. These results supplement those in the main manuscript demonstrating how well we are able to recover the oracle Laplacian spectral decomposition, both the basis and associated eigenvalues, with our direct neural method without any knowledge on the underlying mesh structure. While extremely

[6]Using the ground-truth mesh information that our point cloud method does not have access to.

Table 3. Average cosine similarity between predicted and oracle eigenfunctions at different truncation levels k, and mean relative eigenvalue discrepancy (overfitting setting).

Shape	Image	$k \leq 10$	$k \leq 20$	$k \leq 50$	λ Discrepancy
Beetle		0.855	0.657	0.450	0.514 ± 0.706
David		0.803	0.879	0.823	0.098 ± 0.126
Dente		0.981	0.976	0.910	0.106 ± 0.150
Dragon		0.879	0.827	0.529	0.197 ± 0.481
Eros		0.952	0.891	0.640	0.083 ± 0.156
Heptoroid		0.796	0.757	0.509	0.182 ± 0.185
Horse		0.927	0.908	0.718	0.107 ± 0.128
Knot		0.536	0.687	0.504	0.127 ± 0.279
Wrench		0.938	0.817	0.476	0.694 ± 0.222
Master Cylinder		0.948	0.923	0.779	0.051 ± 0.045
Teddy		0.987	0.981	0.912	0.109 ± 0.146

similar at low frequencies, differences emerge at higher frequencies, revealing that our method can account for different details from the oracle, while still preserving the same overall (i.e. low-pass) information of the shape manifold. Unlike the eigenfunctions, recovering the same eigenvalues as those of the oracle Laplacian is trickier and often requires carefully tuning hyperparameters to generate the appropriate probe functions at test time.

Generelization setting. In Tab. 5, we provide quantitative results comparing our predicted and oracle eigenfunctions on each evaluation dataset. Our generalization method quantitatively demonstrates potential for designing a foundation model to estimate Laplacian spectral bases. A proper foundational model would be achieved by greatly scaling up the amount of training data, as is usually done for foundation models in other fields like image or text processing. In our limited training setting, yet still large for initial insights, our model is able to accurately estimate the first Fourier eigenfunctions, which corresponds to low-pass information and thus to the most important global information on the manifold. As expected, performance is naturally lower than in the single-shape overfitting setting.

C.3. High-Dimensional Manifolds

C.3.1. Short Manifold Learning Overview

Manifold learning is a classical task [44, 68, 120, 133] in data analysis and computer vision. It consists in finding low-dimensional representations capturing the manifold structure of high-dimensional data. In practice, methods aim to find very low-dimensional embeddings in $\mathbb{R}^k$, with $k \ll d$ (where d is the original high dimension), by preserving pairwise dissimilarities based on distances. The wide collection of approaches is usually grouped based on whether the targeted preserved structures of the manifold are local [10, 11, 35, 36, 40, 52, 78, 79, 81, 84, 85, 112, 124, 125, 131, 143, 144], global [23, 47, 137, 138], or a combination of both [1, 22, 25, 38, 39, 58, 61, 63, 83, 94, 97, 110, 114–116, 126].

C.3.2. Implementation

For manifold learning experiments on image datasets, we work on the embedded feature manifold provided by pretrained vision models. Thus, each image is mapped to a high-dimensional[7] feature space, and we learn the spectral decomposition of the resulting manifold structure across four benchmark datasets: Caltech256, CIFAR100, Imagenette, and STL-10.

Feature Extraction. We experiment with two pretrained vision models as feature extractors: CLIP[8] producing 512-dimensional embeddings, and DINOv2[9] producing 384-dimensional embeddings. These frozen feature extractors map images to points on a learned manifold, where geometric relationships reflect semantic similarities. The extracted embeddings are used directly as spatial coordinates for kNN graph construction with cosine similarity.

Network Architecture and Hyperparameters. In Tab. 4, we detail the architecture and probe function generation hyperparameters for each experiment. For CLIP features, we use a preprocessing MLP [512 $\rightarrow$ 256], followed by a 6-layer Transformer encoder with 8 attention heads, $d_{\text{model}} = 256$, feedforward dimension of 256, dropout of 0.1, and GELU activations. The output head is [256 $\rightarrow$ 64 $\rightarrow$ 11] with layer normalization. For DINOv2 features, we use a more compact architecture: preprocessing MLP [384 $\rightarrow$ 128], 6-layer Transformer with 4-8 attention heads depending on the dataset, $d_{\text{model}} = 128$, and a simpler output head [128 $\rightarrow$ 11].

Training Configuration. We optimize with AdamW (learning rate 0.0005) and accumulate gradients over 16 batches. Gradient clipping is applied at 0.01. Training uses distributed data parallel (DDP) [74] strategy across multiple GPUs. Each dataset samples 5000 images uniformly, with deterministic sampling for validation and random sampling for training. We predict 11 eigenvectors for spectral clustering evaluation at $k \in \{2, 5, 10\}$. We generate 300 probe functions per manifold by sampling uniform random values in [-1, 1] at each point.

[7] Yet lower-dimensional than the original image dimensions.

[8] https://huggingface.co/openai/clip-vit-base-patch32

[9] https://huggingface.co/facebook/dinov2-small

Evaluation Setting. We compare our method against classical manifold learning approaches including PCA, UMAP, t-SNE, Isomap, and Laplacian Eigenmaps. For evaluation, we use class-weighted sampling to create challenging, non-uniform manifold distributions. Specifically, we sample 1500 images using random class weights with entropy in the range [0.01, 0.1], ensuring that the sampled manifolds have imbalanced class distributions. This sampling strategy tests the robustness of each method to realistic, non-uniform data distributions across image classes. Due to the randomness of the sampling and of some of the methods, including ours due to random probe functions, we perform 32 runs for each method on each dataset.

Sensitivity to Hyperparameters. Similar to the single-shape overfitting experiments in the 3D case, we observe that results in the image manifold setting are sensitive to the choice of probe function smoothing hyperparameters (k, σ, iterations). The parameters in Table 4 were tuned per dataset to achieve good performance. In future work, these probe function parameters could potentially be learned during training rather than manually tuned. We emphasize that the current results showcase the potential of our method, but more optimized tuning should yield better results. As such, we believe the full potential of our approach for image manifold learning has yet to be realized.

Computational Resources. All experiments were conducted on 8 GPUs (type NVIDIA A100-SXM4-40GB) using PyTorch Lightning with DDP training strategy.

Table 4. Architecture and probe function generation hyperparameters for image manifold experiments. All experiments use 6 Transformer layers, 11 predicted eigenvectors, 5000 sampled images, and 300 probe functions. The kNN graph always uses cosine similarity with self-loops enabled.

Dataset	Feature Extractor	d_{model}	Attn. Heads	k	σ	Iterations
Caltech256	CLIP	256	8	10	0.04	20
Caltech256	DINOv2	128	8	15	0.15	40
CIFAR100	CLIP	256	8	5	0.04	20
CIFAR100	DINOv2	128	4	30	0.20	30
Imagenette	CLIP	256	8	20	0.08	20
Imagenette	DINOv2	128	4	30	0.10	30
STL-10	CLIP	256	8	35	0.08	20
STL-10	DINOv2	128	4	20	0.10	20

C.3.3. Results

We evaluate our method on four image datasets (Caltech256, CIFAR100, Imagenette, and STL-10) using both CLIP and DINOv2 embeddings, comparing against classical manifold learning methods: PCA, UMAP, t-SNE, Isomap, and Laplacian Eigenmaps. Results are reported for spectral clustering with $k \in \{2, 5, 10\}$ clusters across seven standard clustering metrics: Normalized Mutual Information (NMI) [132], Adjusted Rand Index (ARI) [60], Completeness [108], Adjusted Mutual Information (AMI) [132], Homogeneity [108], V-Measure [108], and Fowlkes-Mallows Index (FMI) [45].

Quantitative Results. In Tabs. 6 to 13 we present comprehensive results, averaged over our 32 runs, across all datasets and feature extractors. Note that UMAP and t-SNE often get better clustering scores, which is unsurprising as these methods are designed to handle classification datasets by overclustering their embeddings (better scores yet often artificial separation between clusters). Our method (Optimal Approximation Eigenmaps) demonstrates competitive performance across most settings. Our approach generally outperforms classical methods like PCA, Isomap, and most importantly, shows superior performance to its vanilla analog, Laplacian Eigenmaps, in most cases. On CIFAR100 with CLIP embeddings, our method achieves particularly strong results at $k = 10$ (NMI $= 0.641$, ARI $= 0.257$), matching or exceeding Laplacian Eigenmaps while substantially outperforming PCA and Isomap. Similarly, on Caltech256 with DINOv2 features at $k = 5$, we achieve NMI $= 0.714$ and ARI $= 0.246$, demonstrating effective learned spectral representations. The method shows consistent improvements as the number of clusters increases from 2 to 10, suggesting that higher-dimensional embeddings better capture the manifold structure.

Qualitative Analysis. In Figs. 23 to 30, we plot the 2D (i.e. $k = 2$) clustering results for Imagenette and STL-10 datasets with both CLIP and DINOv2 embeddings. These visualizations demonstrate that our learned spectral basis captures meaningful semantic structure in the image manifolds, with distinct clusters corresponding to different object categories that are better clustered than in traditional methods like PCA and Isomap. In addition to accurate clusters, our embeddings provide smooth transitions between clusters, indicating that the learned basis successfully preserves the underlying smooth manifold geometry, unlike methods artificially overclustering the data like t-SNE and UMAP. This important observation demonstrates that our method is in fact superior to modern references t-SNE and UMAP, which cluster very well as revealed by the quantitative results but poorly preserve the manifold structure. Our visualisations are significantly better than those of Laplacian Eigenmaps, the vanilla analog of our method using the Graph Laplacian eigenfunctions instead of our Optimal Approximation Operator's eigenfunctions. Indeed, in Laplacian Eigenmaps clusters are ill-shaped and artificially separated unlike in our method.

C.4. Generality, adaptivity, and sensitivity of our method

C.4.1. Point cloud size

We emphasize that our method is not constrained to a fixed point cloud size. Indeed, our transformer, which forms the backbone of the method, naturally handles variable-length sequences. While we used a fixed size during training to get efficient batching, the model generalizes to different point counts at inference. For instance, for 3D data, the same overfitted Pegaso model (trained on 4k points) achieves mean cosine similarity 0.606 at 3k points and 0.661 at 12k points. See Fig. 7a for the plot of the 12th basis function inferred on both point clouds of the same Pegaso shape. The adaptivity to sample size also holds for high-dimensional manifolds. Indeed, our results in Figs. 8 and 9 were achieved on models trained on 5000 image samples yet evaluated on 1500. As such, the generalization and robustness to point cloud size at inference holds across all tested shapes and data.

C.4.2. Probe function family

Our default probe function distribution is given by smooth functions obtained by Gaussian smoothing of random functions on the k-nearest neighbor graph. This distribution yields spectral bases resembling those of the Laplacian, which is the default operator in spectral analysis. Nevertheless, our method is not constrained to this probe distribution nor this operator. We show how the resulting spectral basis evolves when working with other probe distributions.

Piecewise-constant probes. To mimic semantic priors, e.g. part labels, we defined probes as random piecewise-constant functions on K clusters. We observe two regimes. When predicting fewer spectral functions than clusters ($k < K$), the network converges to a Walsh-Hadamard-like basis (see Fig. 7b). However, when $k = K$, the spectral basis converges to cluster indicator functions.

Polynomial probes. When the probes are taken to be random 3D-multivariate polynomials of degree at most 2, which is a 10-dimensional space spanned by $\{1, X, Y, Z, X^2, Y^2, Z^2, XY, XZ, YZ\}$, the network converges to a basis spanning the same polynomial subspace (99.93% variance explained vs. 0.24% for a random baseline), despite polynomials being smooth like Laplacian eigenfunctions. As such, smoothness-only is not sufficient to get a spectral basis resembling that of the Laplacian, as we also crucially require that the distribution of smooth functions to be uniform. Unlike uniformly random polynomials, Gaussian-smoothed uniformly random probes approximately uniformly sample from the Laplacian's low-energy eigenspace.

Schrödinger-like probes. When modulating probes by a potential proportional to Gaussian curvature before smoothing, the recovered basis resembles eigenfunctions of the Hamiltonian $H = \Delta + \beta V$ (Fig. 7c), which is the Laplacian's natural generalization with a potential.

These experiments confirm that different probe families yield different spectral bases, each having their own properties and thus advantages for downstream tasks. Our default probe distribution gives a Laplacian-like spectral basis, which is the most useful family of basis in general for spectral analysis. Additionally, designing families of probe distributions is a fairly easy task when compared to discretizing operators, especially in high-dimensions. Passing the burden from the choice of operator to that of the distribution of probes is thus a significant advantage.

C.4.3. Hyperparameter sensitivity

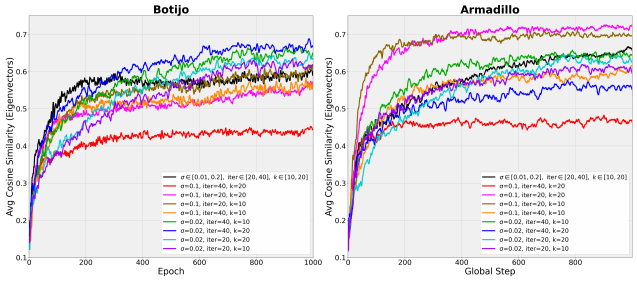

Figure 10. Hyperparameter sensitivity.

Our method does not necessarily require precise hyperparameter tuning of the probe distribution. Indeed, in our 3D experiments we use identical parameters across all shapes. Sensitivity primarily appears for recovering specific eigenvalues, but not for estimating the spectral basis. In Fig. 10, we show that different hyperparameters all yield bases closely resembling cotangent Laplacian eigenfunctions. In fact, a "wildcard" setting, with randomly sampled hyperparameters per probe, achieves competitive cosine similarity. This demonstrates that there is no need for precise hyperparameter tuning when estimating the spectral basis. In Tab. 2, we show that per-shape hyperparameter tuning is required to recover exactly the cotangent Laplacian eigenvalues. Without tuning, the basis still exhibits correct spectral decay (Fig. 22) and may benefit downstream tasks where exact recovery is unnecessary, although we did not test this. Note that probe hyperparameters could also be made learnable with respect to downstream applications.

C.5. Computational cost

In the 3D overfitting setting, it takes approximately 8min with an RTX 3090 GPU to train the model on a shape with 4k points. As overfitted models generalize to other samplings of the same shape with different number of samples (see Appendix C.4.1), it is not necessary to train an overfitting model on many more points if the point cloud is larger. For inference, computing the first 50 spectral basis vectors takes 0.44s/1.14s/4.20s on 3k/12k/24k

points, which is comparable to the robust Laplacian [118] (0.20s/1.24s/2.62s). We did not use an advanced optimized implementation, such as torch.compile, flash attention, or custom CUDA kernels. Thus, inference time could be substantially improved, especially with batching. In the 3D generalization setting, it takes 24h on 8 A100 GPUs to train the model, albeit the model is only trained once (unlike in the overfitting scheme where it must be retrained for each shape). Inference is then simply a single forward pass, identical to the overfitting setting (in particular it can be done on single commercial GPU like the RTX 3090). For high-dimensional manifolds, we use small manifolds of approximately 1500 samples to run on a single RTX 3090 GPU.

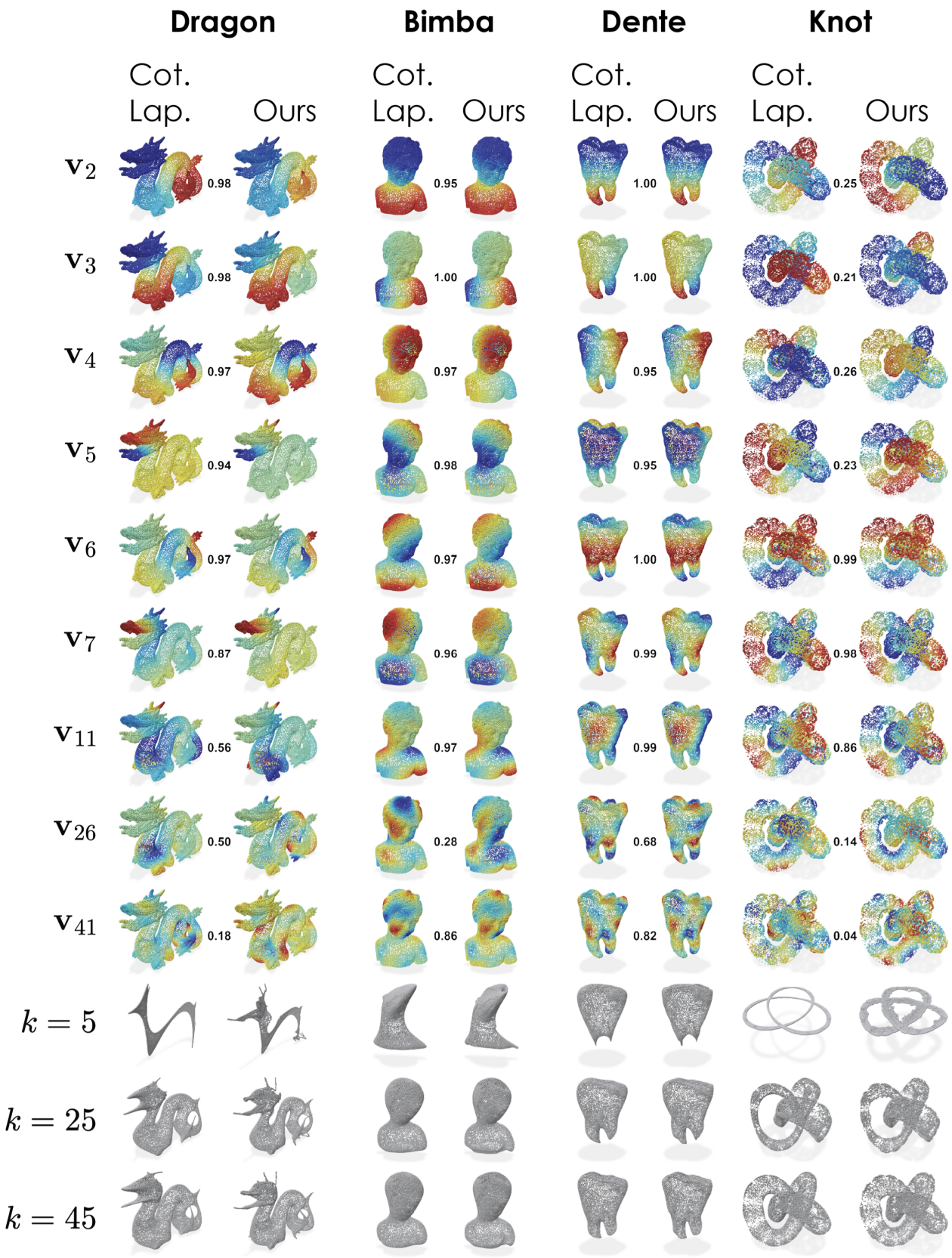

Figure 11. Unnormalized spectral basis (top) and xyz reconstruction from k basis vectors (bottom), using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

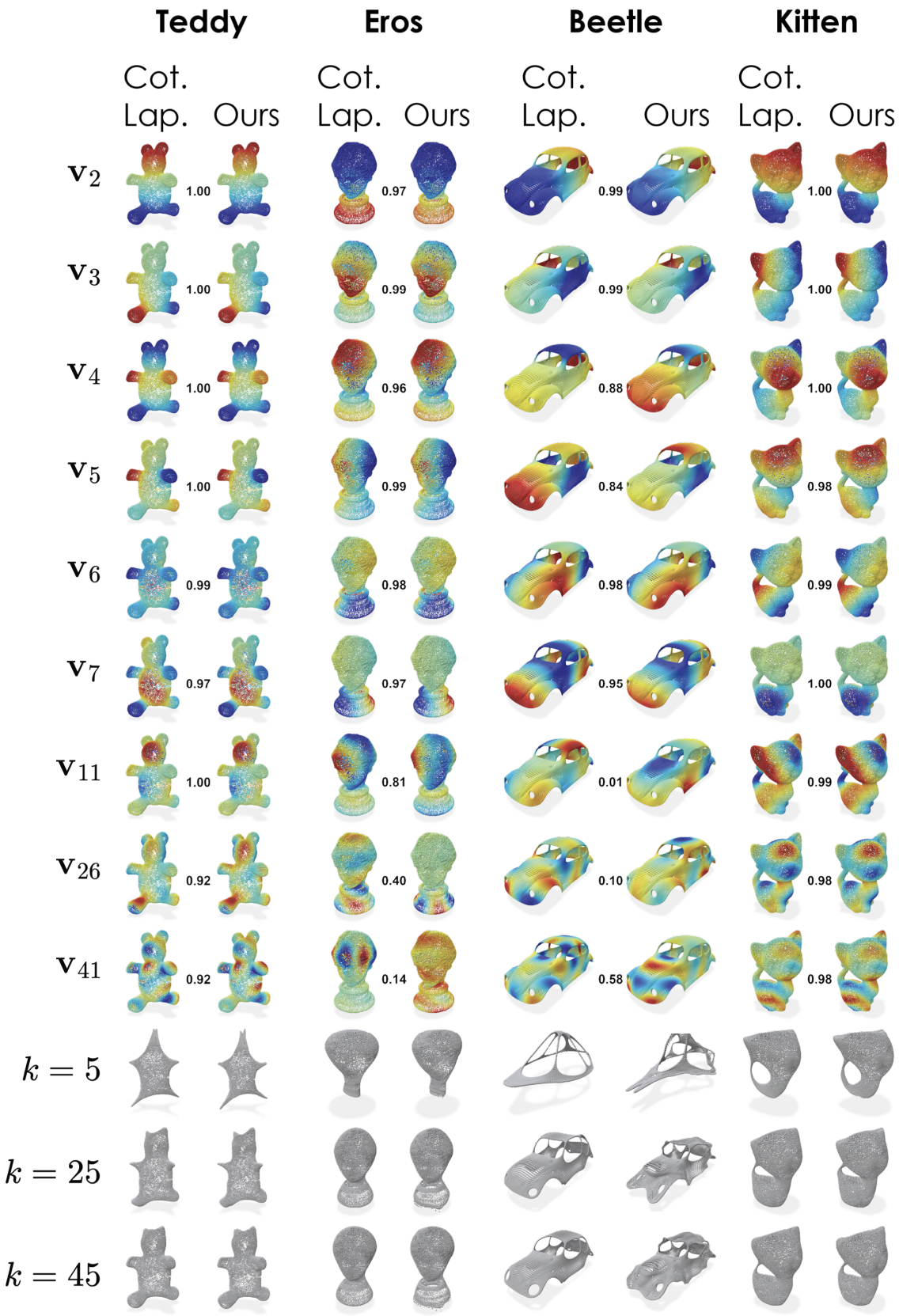

Figure 12. Unnormalized spectral basis (top) and xyz reconstruction from k basis vectors (bottom), using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

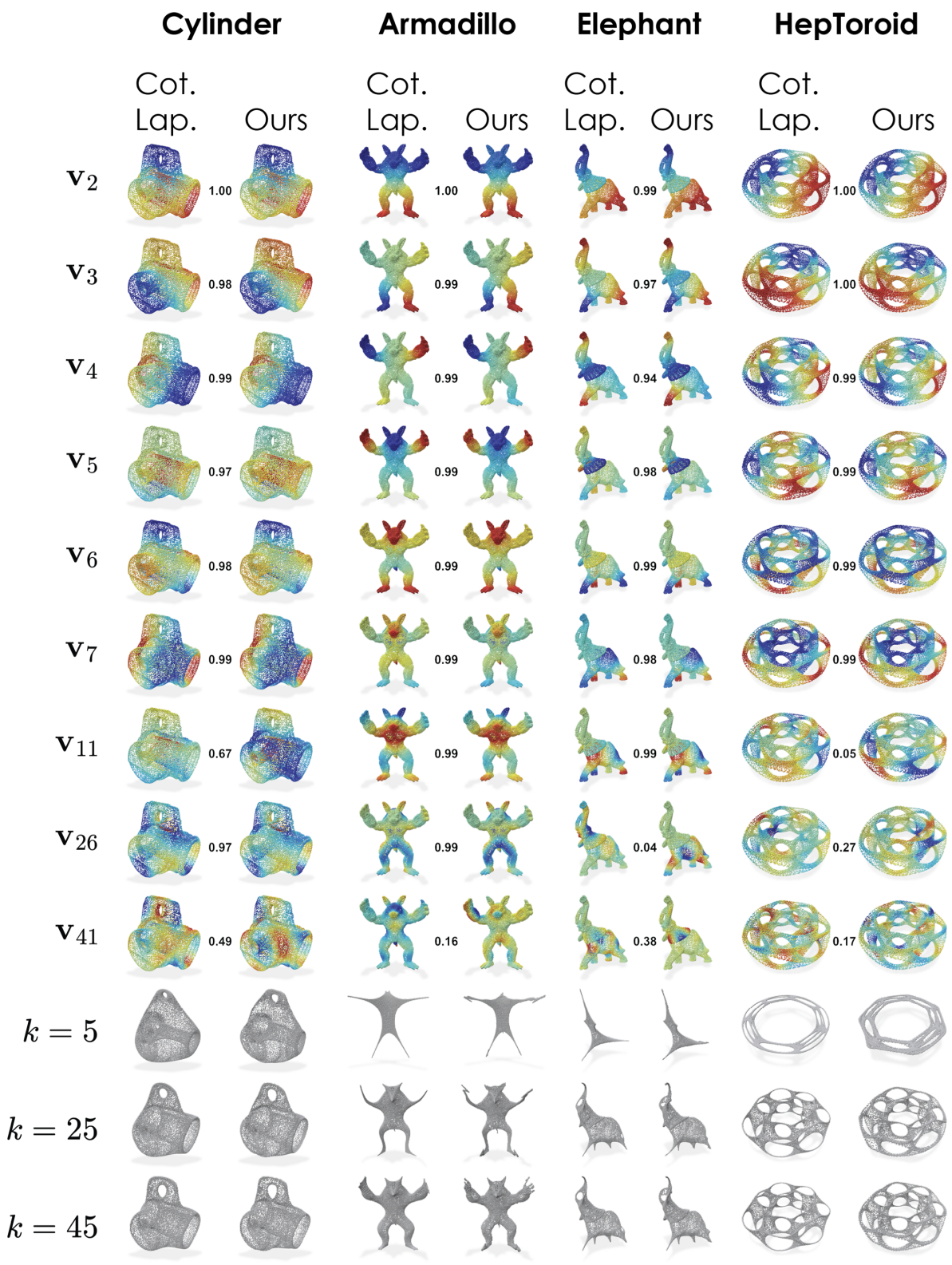

Figure 13

Unnormalized spectral basis (top) and xyz reconstruction from k basis vectors (bottom), using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

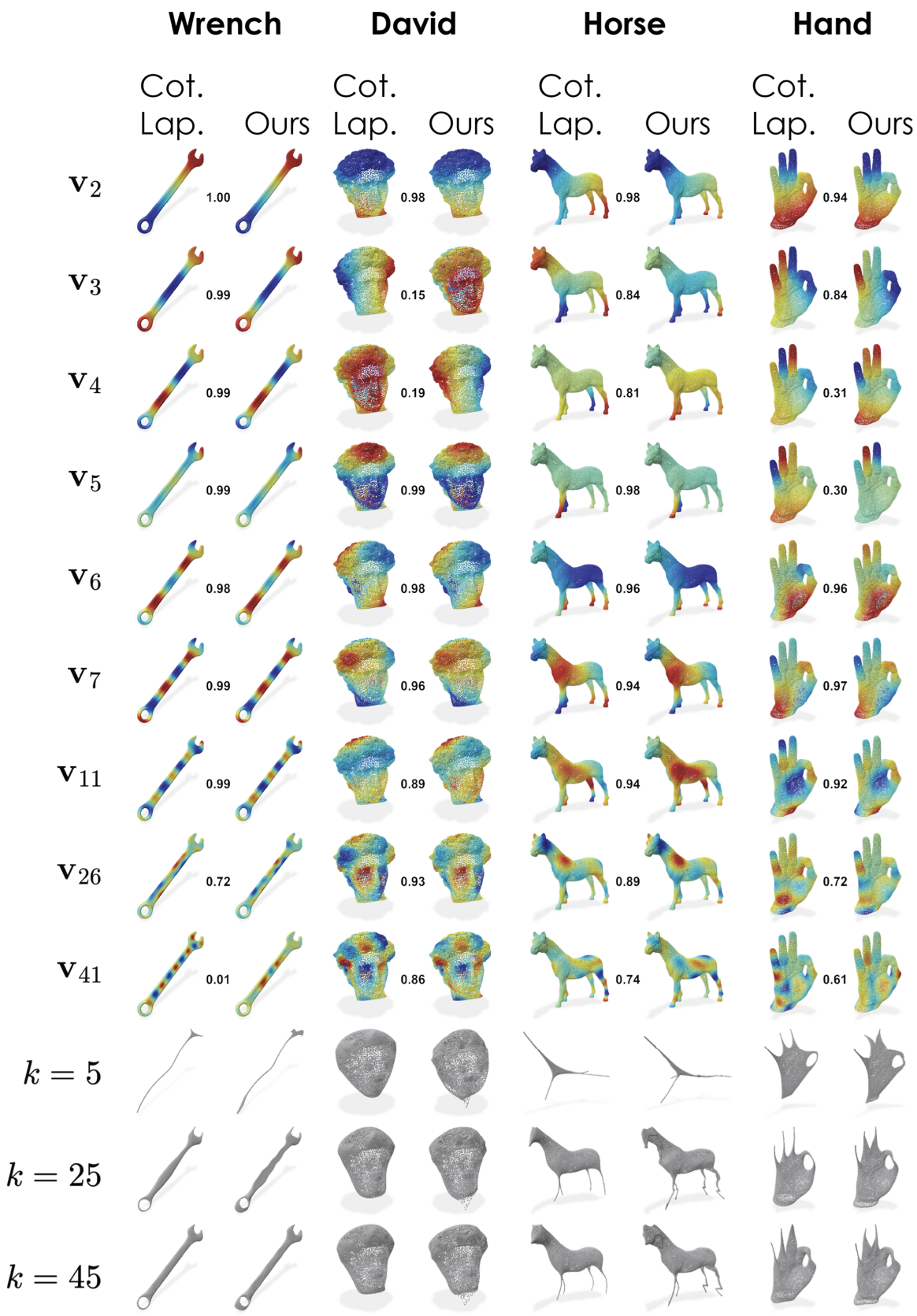

Figure 14. Unnormalized spectral basis (top) and xyz reconstruction from k basis vectors (bottom), using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

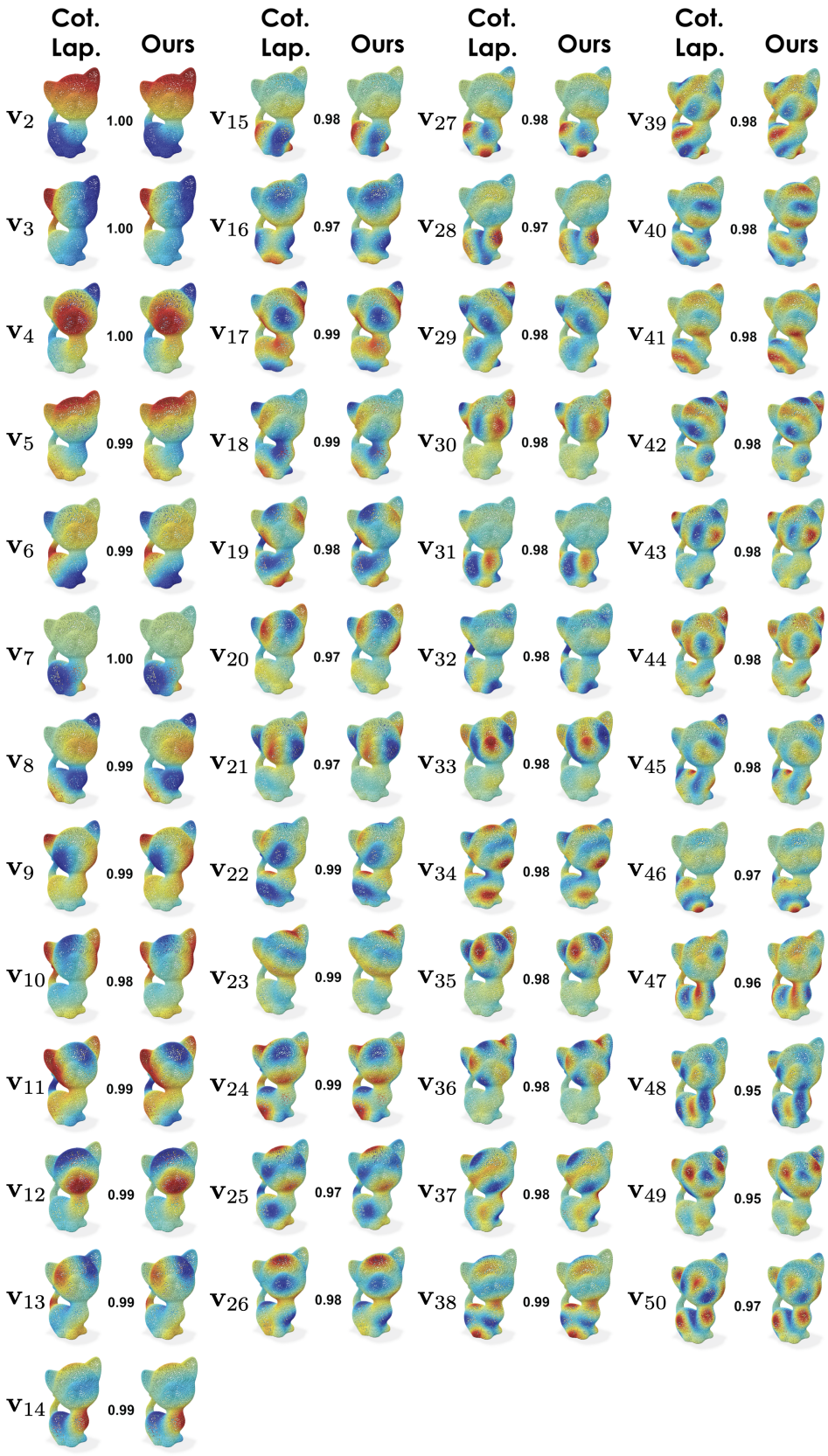

Figure 15. Unnormalized spectral basis using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

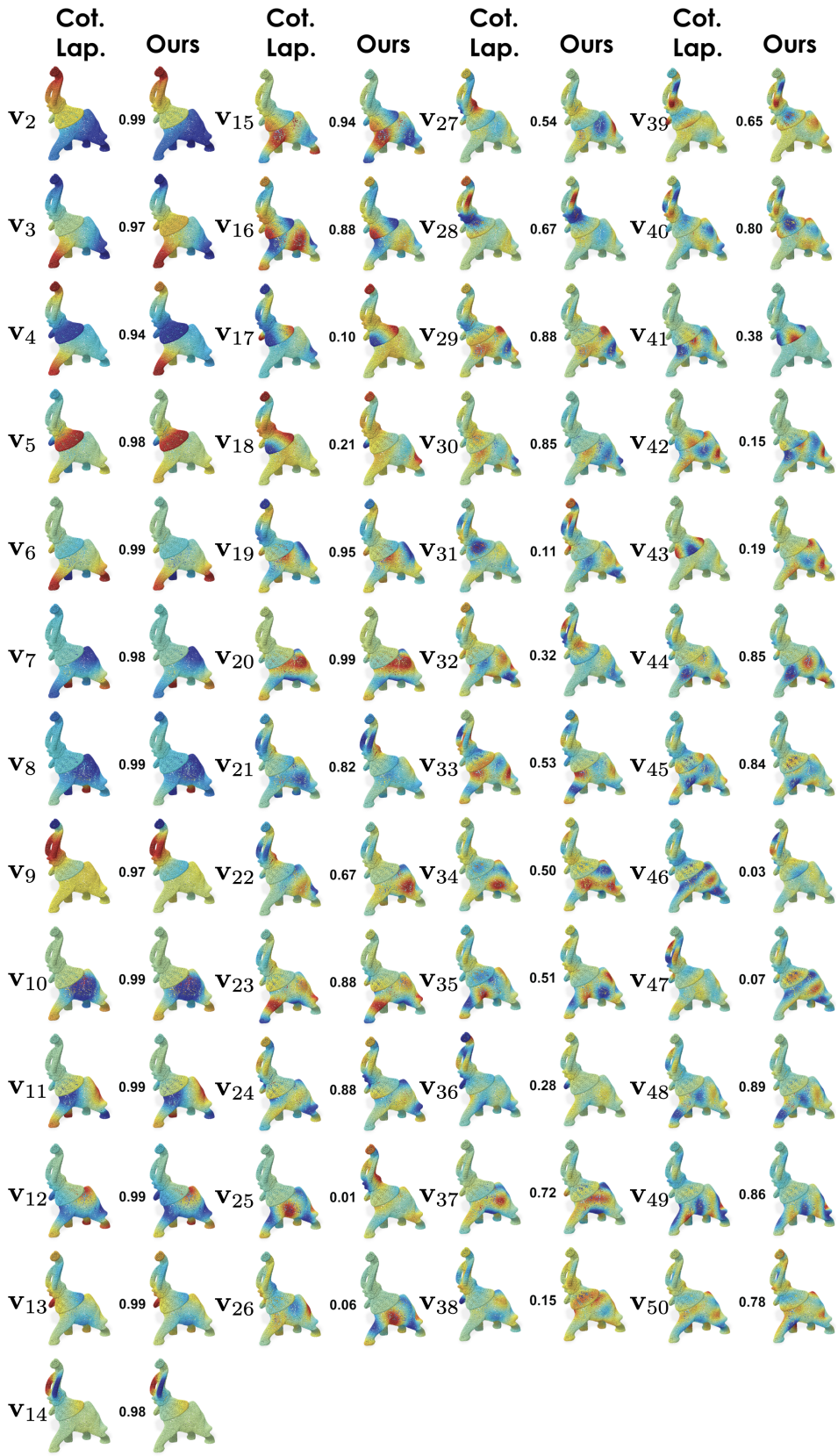

Figure 16. Unnormalized spectral basis using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

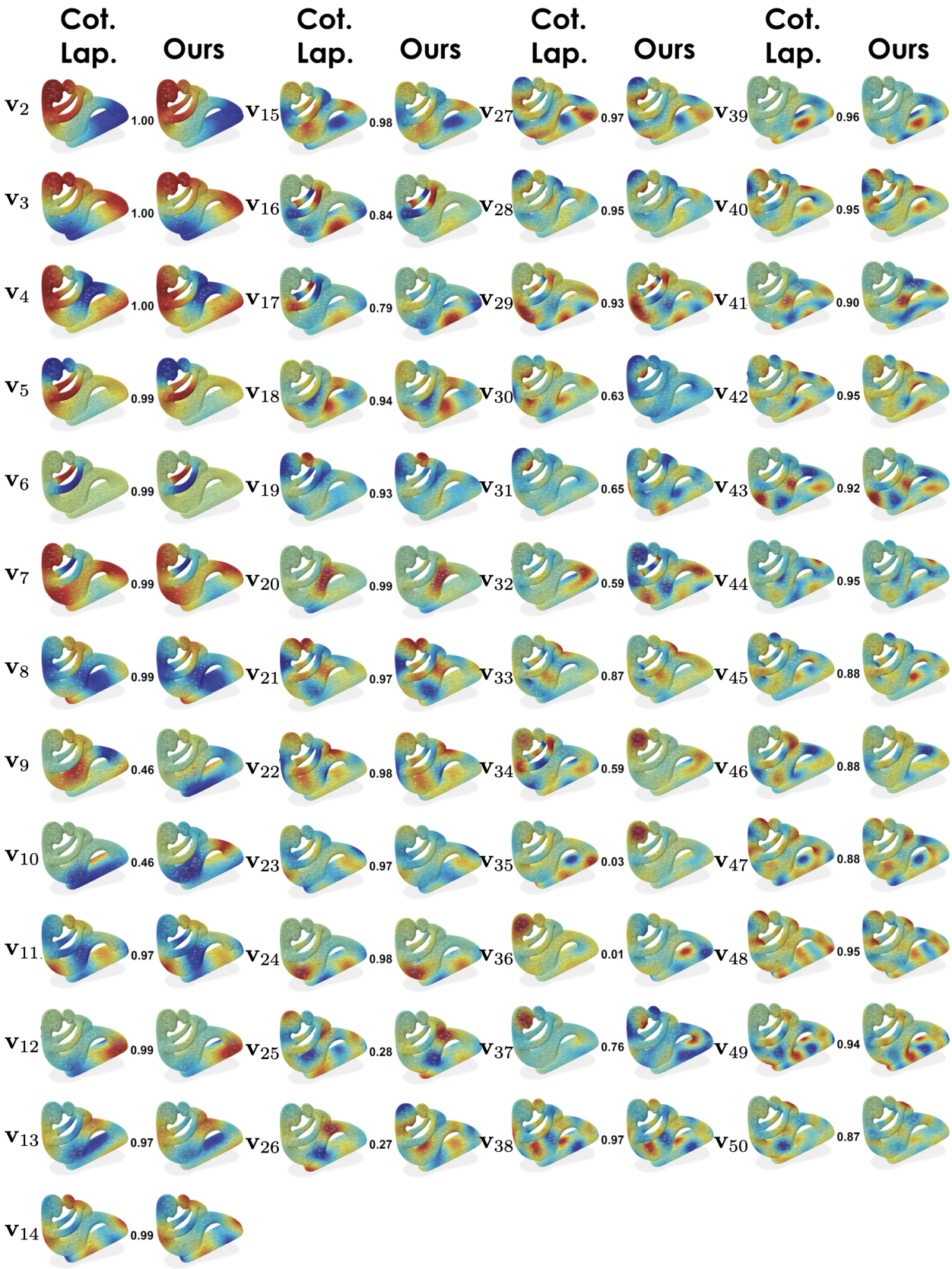

Figure 17. Unnormalized spectral basis using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

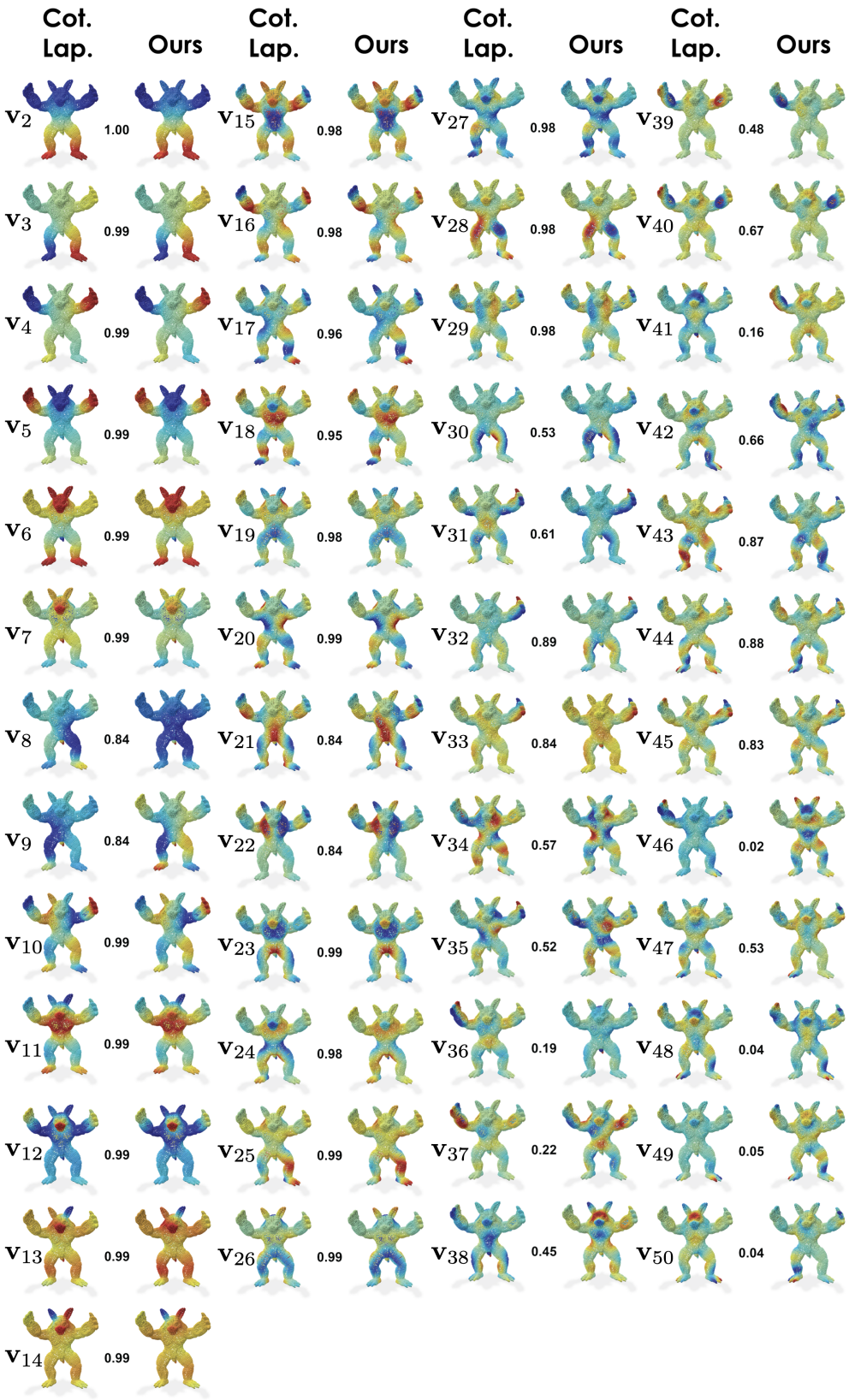

Figure 18. Unnormalized spectral basis using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

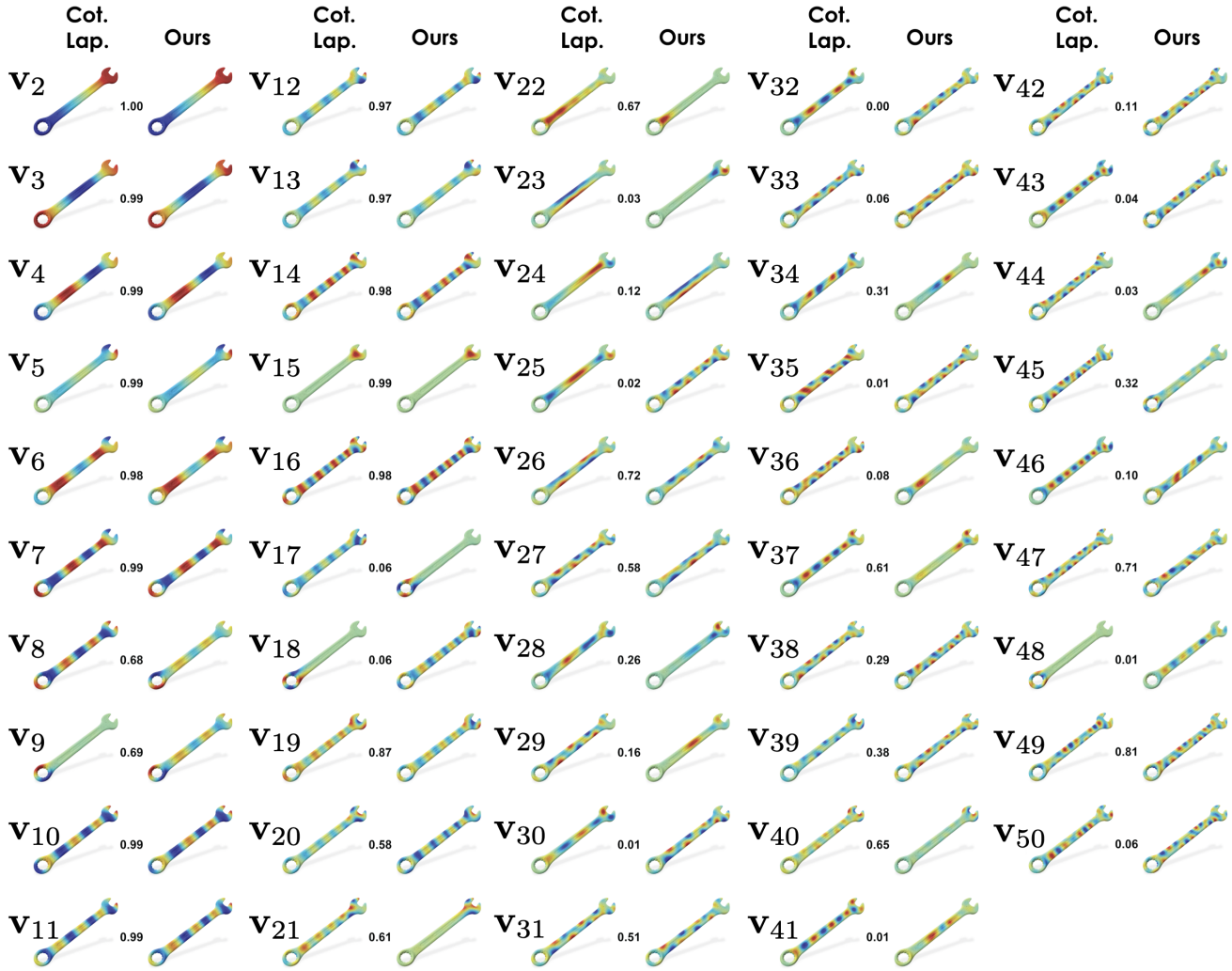

Figure 19. Unnormalized spectral basis using either the oracle cotangent Laplacian or our method (overfitting setting). Scalars are cosine similarities between basis vectors.

Figure 20. Estimated mass metric M from $\mathbf{q}_1$ (overfitting setting).

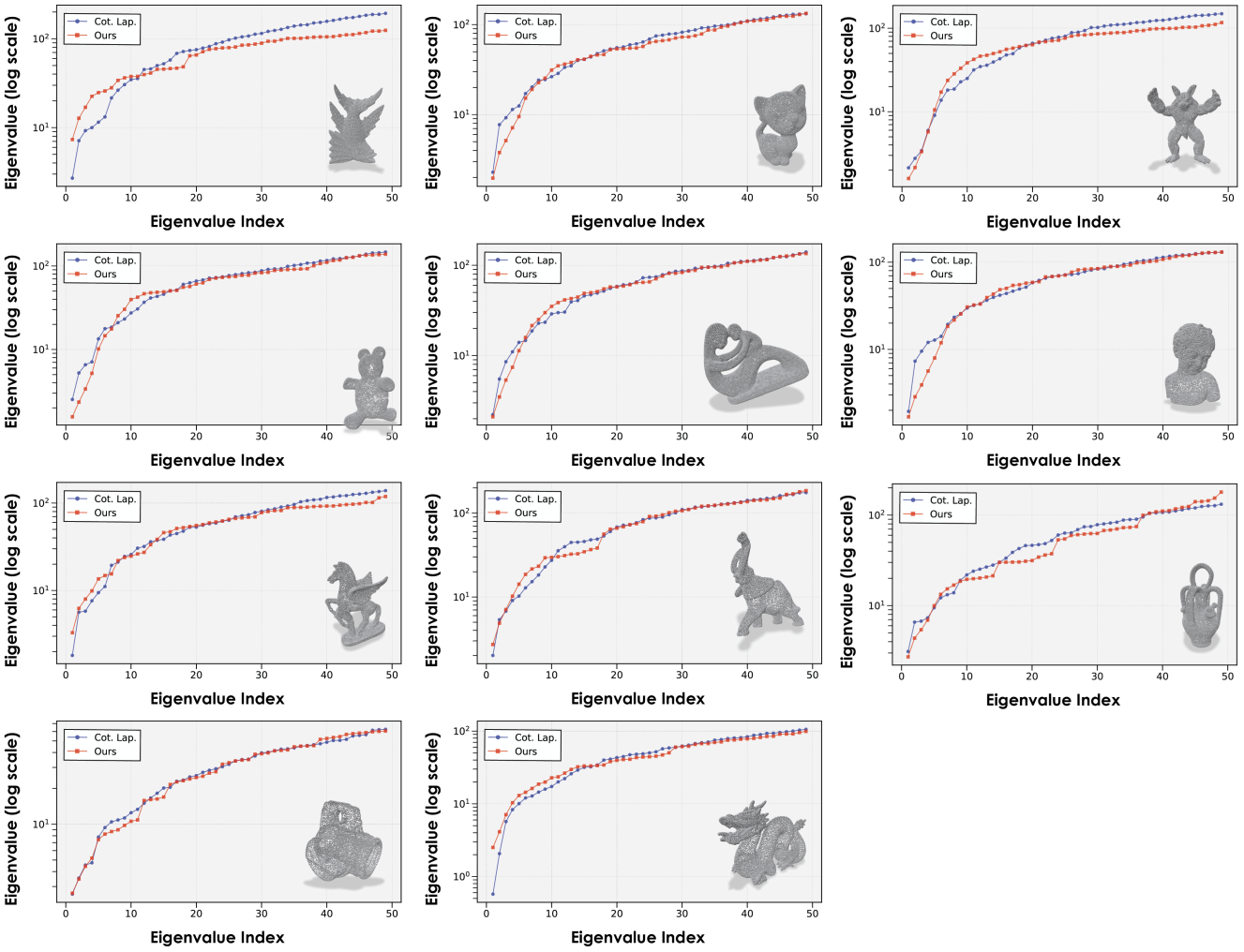

Figure 21. Eigenvalues of the oracle cotangent Laplacian and our estimated ones (overfitting setting, fine-tuned hyperparameter configuration for generating probe functions per shape).

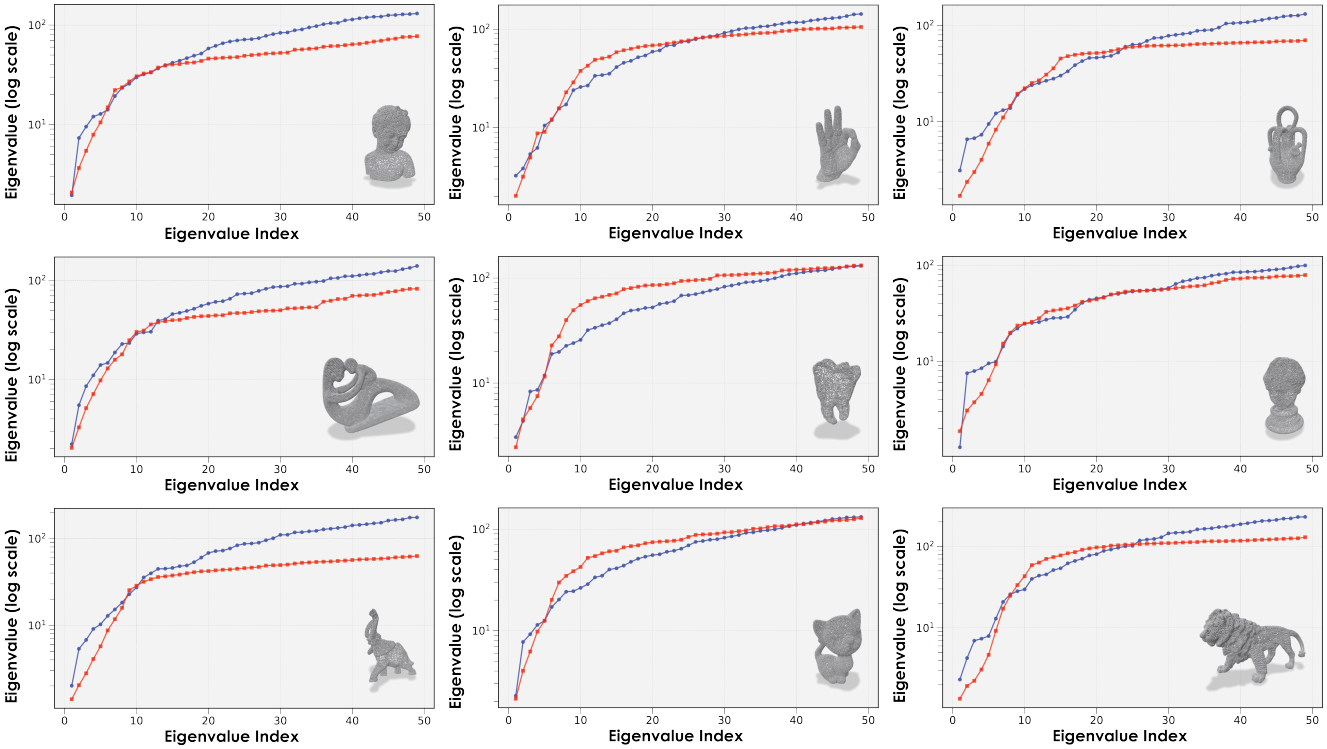

Figure 22. Eigenvalues of the oracle cotangent Laplacian and our estimated ones (overfitting setting, fixed hyperparameter configuration for generating probe functions across all shapes).

Table 5. Average cosine similarity between predicted and oracle eigenfunctions at different truncation levels k on each evaluation dataset (generalization setting).

Dataset	**#Shapes**	$k \leq 5$	$k \leq 10$	$k \leq 15$	$k \leq 20$	$k \leq 25$	$k \leq 30$	$k \leq 35$	$k \leq 40$	$k \leq 45$	$k \leq 50$
MPZ14	105	0.735	0.612	0.526	0.473	0.429	0.393	0.365	0.341	0.320	0.301
SHREC'07	380	0.729	0.610	0.535	0.480	0.438	0.403	0.374	0.349	0.327	0.308
SHREC'14	700	0.793	0.666	0.585	0.532	0.478	0.438	0.404	0.377	0.352	0.329
SHREC'15	1200	0.693	0.570	0.511	0.458	0.417	0.384	0.356	0.331	0.310	0.291
SHREC'19	50	0.750	0.633	0.556	0.495	0.448	0.409	0.381	0.357	0.334	0.315
SHREC'20-GR	220	0.670	0.514	0.435	0.385	0.348	0.321	0.296	0.277	0.261	0.247
SHREC'20-NI	14	0.598	0.536	0.495	0.446	0.404	0.375	0.348	0.326	0.307	0.291
DefTransfer	278	0.621	0.500	0.425	0.386	0.352	0.326	0.303	0.285	0.269	0.254
TopKids	26	0.788	0.656	0.551	0.486	0.438	0.402	0.374	0.352	0.331	0.312

Table 6. Average manifold learning results on STL10 with DINOv2 embeddings. The mean and standard deviation over 32 runs is provided. The best and second best results are in bold and underlined respectively.

STL10 - DINOv2								
Method	k	NMI	ARI	Comp.	AMI	Homo.	V-Meas.	FMI
Opt. App. Eigenmaps (Ours)	2	$\underline{0.790 \pm 0.083}$	$\mathbf{0.712 \pm 0.144}$	$\mathbf{0.752 \pm 0.094}$	$\underline{0.787 \pm 0.084}$	0.834 ± 0.080	$\underline{0.790 \pm 0.083}$	$\mathbf{0.775 \pm 0.110}$
Laplacian Eigenmaps	2	0.594 ± 0.075	0.382 ± 0.096	0.568 ± 0.058	0.589 ± 0.076	0.627 ± 0.111	0.594 ± 0.075	0.510 ± 0.095
PCA	2	0.668 ± 0.046	0.475 ± 0.073	0.611 ± 0.041	0.664 ± 0.046	0.740 ± 0.075	0.668 ± 0.046	0.579 ± 0.076
UMAP	2	$\mathbf{0.808 \pm 0.058}$	$\underline{0.633 \pm 0.119}$	$\underline{0.741 \pm 0.082}$	$\mathbf{0.806 \pm 0.059}$	$\mathbf{0.894 \pm 0.038}$	$\mathbf{0.808 \pm 0.058}$	$\underline{0.717 \pm 0.081}$
t-SNE	2	0.789 ± 0.058	0.595 ± 0.106	0.718 ± 0.080	0.786 ± 0.058	$\underline{0.879 \pm 0.033}$	0.789 ± 0.058	0.686 ± 0.070
Isomap	2	0.684 ± 0.052	0.520 ± 0.092	0.632 ± 0.062	0.681 ± 0.053	0.751 ± 0.066	0.684 ± 0.052	0.618 ± 0.074
Opt. App. Eigenmaps (Ours)	5	$\mathbf{0.859 \pm 0.061}$	$\mathbf{0.812 \pm 0.120}$	$\mathbf{0.823 \pm 0.081}$	$\mathbf{0.858 \pm 0.061}$	$\mathbf{0.902 \pm 0.047}$	$\mathbf{0.859 \pm 0.061}$	$\mathbf{0.856 \pm 0.088}$
Laplacian Eigenmaps	5	0.815 ± 0.054	$\underline{0.734 \pm 0.130}$	$\underline{0.793 \pm 0.063}$	0.813 ± 0.055	0.842 ± 0.065	0.815 ± 0.054	$\underline{0.792 \pm 0.104}$
PCA	5	0.746 ± 0.053	0.565 ± 0.092	0.682 ± 0.073	0.743 ± 0.053	0.828 ± 0.043	0.746 ± 0.053	0.658 ± 0.061
UMAP	5	$\underline{0.821 \pm 0.051}$	0.666 ± 0.109	0.757 ± 0.076	$\underline{0.819 \pm 0.051}$	$\underline{0.901 \pm 0.031}$	$\underline{0.821 \pm 0.051}$	0.743 ± 0.074
t-SNE	5	0.796 ± 0.060	0.616 ± 0.112	0.727 ± 0.083	0.793 ± 0.060	0.884 ± 0.037	0.796 ± 0.060	0.704 ± 0.075
Isomap	5	0.772 ± 0.047	0.626 ± 0.104	0.711 ± 0.071	0.769 ± 0.048	0.849 ± 0.037	0.772 ± 0.047	0.709 ± 0.069
Opt. App. Eigenmaps (Ours)	10	$\mathbf{0.892 \pm 0.047}$	$\mathbf{0.863 \pm 0.098}$	$\mathbf{0.862 \pm 0.071}$	$\mathbf{0.891 \pm 0.048}$	$\mathbf{0.927 \pm 0.029}$	$\mathbf{0.892 \pm 0.047}$	$\mathbf{0.896 \pm 0.071}$
Laplacian Eigenmaps	10	$\underline{0.828 \pm 0.077}$	$\underline{0.768 \pm 0.124}$	$\underline{0.835 \pm 0.068}$	$\underline{0.826 \pm 0.077}$	0.827 ± 0.101	$\underline{0.828 \pm 0.077}$	$\underline{0.829 \pm 0.073}$
PCA	10	0.788 ± 0.045	0.620 ± 0.097	0.721 ± 0.068	0.785 ± 0.046	0.873 ± 0.036	0.788 ± 0.045	0.706 ± 0.065
UMAP	10	0.821 ± 0.051	0.665 ± 0.109	0.758 ± 0.076	0.819 ± 0.051	$\underline{0.901 \pm 0.031}$	0.821 ± 0.051	0.742 ± 0.074
t-SNE	10	0.810 ± 0.051	0.643 ± 0.108	0.743 ± 0.077	0.808 ± 0.051	0.896 ± 0.030	0.810 ± 0.051	0.726 ± 0.071
Isomap	10	0.794 ± 0.050	0.656 ± 0.101	0.733 ± 0.072	0.791 ± 0.050	0.870 ± 0.037	0.794 ± 0.050	0.733 ± 0.068

Table 7. Average manifold learning results on STL10 with CLIP embeddings. The mean and standard deviation over 32 runs is provided. The best and second best results are in bold and underlined respectively.

STL10 - CLIP								
Method	k	NMI	ARI	Comp.	AMI	Homo.	V-Meas.	FMI
Opt. App. Eigenmaps (Ours)	2	0.691 ± 0.071	0.525 ± 0.108	0.653 ± 0.071	0.687 ± 0.072	0.736 ± 0.085	0.691 ± 0.071	0.622 ± 0.093
Laplacian Eigenmaps	2	0.585 ± 0.071	0.365 ± 0.109	0.558 ± 0.056	0.580 ± 0.073	0.621 ± 0.112	0.585 ± 0.071	0.493 ± 0.110
PCA	2	0.673 ± 0.041	0.515 ± 0.071	0.615 ± 0.043	0.669 ± 0.042	0.746 ± 0.063	0.673 ± 0.041	0.613 ± 0.069
UMAP	2	$\mathbf{0.835 \pm 0.050}$	$\mathbf{0.676 \pm 0.111}$	$\mathbf{0.770 \pm 0.078}$	$\mathbf{0.833 \pm 0.051}$	$\mathbf{0.915 \pm 0.028}$	$\mathbf{0.835 \pm 0.050}$	$\mathbf{0.752 \pm 0.075}$
t-SNE	2	$\underline{0.801 \pm 0.056}$	$\underline{0.614 \pm 0.110}$	$\underline{0.730 \pm 0.080}$	$\underline{0.798 \pm 0.057}$	$\underline{0.892 \pm 0.032}$	$\underline{0.801 \pm 0.056}$	$\underline{0.703 \pm 0.072}$
Isomap	2	0.718 ± 0.053	0.590 ± 0.113	0.667 ± 0.065	0.715 ± 0.053	0.781 ± 0.060	0.718 ± 0.053	0.675 ± 0.095
Opt. App. Eigenmaps (Ours)	5	$\underline{0.820 \pm 0.060}$	$\mathbf{0.741 \pm 0.120}$	$\mathbf{0.795 \pm 0.072}$	$\underline{0.818 \pm 0.060}$	0.850 ± 0.063	$\underline{0.820 \pm 0.060}$	$\mathbf{0.797 \pm 0.094}$
Laplacian Eigenmaps	5	0.796 ± 0.042	$\underline{0.716 \pm 0.107}$	0.774 ± 0.063	0.794 ± 0.043	0.823 ± 0.045	0.796 ± 0.042	$\underline{0.775 \pm 0.095}$
PCA	5	0.760 ± 0.043	0.593 ± 0.091	0.694 ± 0.061	0.757 ± 0.044	0.844 ± 0.041	0.760 ± 0.043	0.682 ± 0.066
UMAP	5	$\mathbf{0.838 \pm 0.049}$	0.694 ± 0.103	$\underline{0.777 \pm 0.074}$	$\mathbf{0.837 \pm 0.050}$	$\mathbf{0.914 \pm 0.028}$	$\mathbf{0.838 \pm 0.049}$	0.765 ± 0.070
t-SNE	5	0.803 ± 0.064	0.632 ± 0.118	0.735 ± 0.088	0.801 ± 0.064	$\underline{0.890 \pm 0.036}$	0.803 ± 0.064	0.717 ± 0.080
Isomap	5	0.796 ± 0.041	0.672 ± 0.102	0.741 ± 0.063	0.794 ± 0.042	0.865 ± 0.032	0.796 ± 0.041	0.745 ± 0.077
Opt. App. Eigenmaps (Ours)	10	$\mathbf{0.863 \pm 0.047}$	$\mathbf{0.818 \pm 0.086}$	$\underline{0.845 \pm 0.063}$	$\mathbf{0.861 \pm 0.047}$	0.884 ± 0.047	$\mathbf{0.863 \pm 0.047}$	$\mathbf{0.858 \pm 0.067}$
Laplacian Eigenmaps	10	0.796 ± 0.131	$\underline{0.720 \pm 0.166}$	$\mathbf{0.850 \pm 0.055}$	0.794 ± 0.133	0.770 ± 0.161	0.796 ± 0.131	$\underline{0.803 \pm 0.086}$
PCA	10	0.788 ± 0.051	0.628 ± 0.103	0.723 ± 0.072	0.785 ± 0.051	0.870 ± 0.035	0.788 ± 0.051	0.711 ± 0.073
UMAP	10	$\underline{0.833 \pm 0.049}$	0.685 ± 0.103	0.772 ± 0.074	$\underline{0.832 \pm 0.049}$	$\mathbf{0.910 \pm 0.029}$	$\underline{0.833 \pm 0.049}$	0.758 ± 0.070
t-SNE	10	0.811 ± 0.066	0.650 ± 0.118	0.745 ± 0.090	0.809 ± 0.067	$\underline{0.894 \pm 0.038}$	0.811 ± 0.066	0.731 ± 0.080
Isomap	10	0.814 ± 0.047	0.697 ± 0.111	0.759 ± 0.069	0.812 ± 0.048	0.881 ± 0.031	0.814 ± 0.047	0.765 ± 0.082

Table 8. Average manifold learning results on Imagenette with DINOv2 embeddings. The mean and standard deviation over 32 runs is provided. The best and second best results are in bold and underlined respectively.

Imagenette - DINOv2								
Method	k	NMI	ARI	Comp.	AMI	Homo.	V-Meas.	FMI
Opt. App. Eigenmaps (Ours)	2	0.684 ± 0.061	0.602 ± 0.114	0.655 ± 0.070	0.680 ± 0.062	0.716 ± 0.053	0.684 ± 0.061	0.667 ± 0.098
Laplacian Eigenmaps	2	0.642 ± 0.062	0.456 ± 0.102	0.627 ± 0.048	0.638 ± 0.063	0.659 ± 0.082	0.642 ± 0.062	0.550 ± 0.091
PCA	2	0.645 ± 0.075	0.486 ± 0.126	0.617 ± 0.062	0.641 ± 0.077	0.677 ± 0.092	0.645 ± 0.075	0.568 ± 0.113
UMAP	2	$\mathbf{0.875 \pm 0.035}$	$\mathbf{0.802 \pm 0.076}$	$\mathbf{0.836 \pm 0.053}$	$\mathbf{0.873 \pm 0.036}$	$\mathbf{0.918 \pm 0.023}$	$\mathbf{0.875 \pm 0.035}$	$\mathbf{0.839 \pm 0.060}$
t-SNE	2	$\underline{0.829 \pm 0.036}$	$\underline{0.716 \pm 0.072}$	$\underline{0.782 \pm 0.045}$	$\underline{0.828 \pm 0.036}$	$\underline{0.884 \pm 0.036}$	$\underline{0.829 \pm 0.036}$	$\underline{0.767 \pm 0.058}$
Isomap	2	0.679 ± 0.045	0.599 ± 0.090	0.646 ± 0.041	0.676 ± 0.046	0.717 ± 0.058	0.679 ± 0.045	0.665 ± 0.080
Opt. App. Eigenmaps (Ours)	5	$\underline{0.842 \pm 0.043}$	$\mathbf{0.844 \pm 0.051}$	$\underline{0.818 \pm 0.058}$	$\underline{0.840 \pm 0.044}$	0.868 ± 0.032	$\underline{0.842 \pm 0.043}$	$\mathbf{0.872 \pm 0.041}$
Laplacian Eigenmaps	5	0.800 ± 0.069	0.703 ± 0.137	0.787 ± 0.071	0.798 ± 0.070	0.817 ± 0.081	0.800 ± 0.069	0.754 ± 0.117
PCA	5	0.759 ± 0.048	0.629 ± 0.080	0.718 ± 0.049	0.756 ± 0.048	0.806 ± 0.056	0.759 ± 0.048	0.691 ± 0.070
UMAP	5	$\mathbf{0.881 \pm 0.038}$	$\underline{0.816 \pm 0.086}$	$\mathbf{0.845 \pm 0.058}$	$\mathbf{0.880 \pm 0.039}$	$\mathbf{0.922 \pm 0.021}$	$\mathbf{0.881 \pm 0.038}$	$\underline{0.850 \pm 0.068}$
t-SNE	5	0.825 ± 0.029	0.725 ± 0.059	0.779 ± 0.040	0.823 ± 0.029	$\underline{0.879 \pm 0.030}$	0.825 ± 0.029	0.775 ± 0.046
Isomap	5	0.836 ± 0.039	0.790 ± 0.077	0.802 ± 0.049	0.834 ± 0.040	0.875 ± 0.045	0.836 ± 0.039	0.827 ± 0.065
Opt. App. Eigenmaps (Ours)	10	0.858 ± 0.046	$\mathbf{0.856 \pm 0.055}$	$\underline{0.832 \pm 0.060}$	0.857 ± 0.046	0.888 ± 0.037	0.858 ± 0.046	$\mathbf{0.881 \pm 0.044}$
Laplacian Eigenmaps	10	0.752 ± 0.092	0.577 ± 0.145	0.820 ± 0.061	0.749 ± 0.092	0.706 ± 0.122	0.752 ± 0.092	0.685 ± 0.091
PCA	10	0.795 ± 0.047	0.676 ± 0.080	0.754 ± 0.052	0.793 ± 0.047	0.841 ± 0.052	0.795 ± 0.047	0.732 ± 0.067
UMAP	10	$\mathbf{0.882 \pm 0.038}$	$\underline{0.818 \pm 0.086}$	$\mathbf{0.846 \pm 0.058}$	$\mathbf{0.881 \pm 0.039}$	$\mathbf{0.923 \pm 0.021}$	$\mathbf{0.882 \pm 0.038}$	0.851 ± 0.068
t-SNE	10	0.820 ± 0.036	0.721 ± 0.068	0.776 ± 0.047	0.818 ± 0.037	0.871 ± 0.034	0.820 ± 0.036	0.771 ± 0.055
Isomap	10	$\underline{0.862 \pm 0.043}$	0.818 ± 0.093	0.828 ± 0.063	$\underline{0.860 \pm 0.043}$	$\underline{0.901 \pm 0.030}$	$\underline{0.862 \pm 0.043}$	$\underline{0.851 \pm 0.073}$

Table 9. Average manifold learning results on Imagenette with CLIP embeddings. The mean and standard deviation over 32 runs is provided. The best and second best results are in bold and underlined respectively.

Imagenette - CLIP								
Method	k	NMI	ARI	Comp.	AMI	Homo.	V-Meas.	FMI
Opt. App. Eigenmaps (Ours)	2	0.746 ± 0.055	0.680 ± 0.083	0.712 ± 0.061	0.743 ± 0.056	0.784 ± 0.056	0.746 ± 0.055	0.733 ± 0.069
Laplacian Eigenmaps	2	0.519 ± 0.109	0.281 ± 0.130	0.532 ± 0.083	0.513 ± 0.112	0.515 ± 0.140	0.519 ± 0.109	0.421 ± 0.104
PCA	2	0.682 ± 0.043	0.551 ± 0.073	0.642 ± 0.034	0.678 ± 0.044	0.728 ± 0.060	0.682 ± 0.043	0.623 ± 0.071
UMAP	2	$\mathbf{0.895 \pm 0.042}$	$\mathbf{0.818 \pm 0.095}$	$\mathbf{0.855 \pm 0.065}$	$\mathbf{0.894 \pm 0.042}$	$\mathbf{0.941 \pm 0.017}$	$\mathbf{0.895 \pm 0.042}$	$\mathbf{0.853 \pm 0.074}$
t-SNE	2	$\underline{0.876 \pm 0.041}$	$\underline{0.782 \pm 0.091}$	$\underline{0.831 \pm 0.063}$	$\underline{0.875 \pm 0.042}$	$\underline{0.929 \pm 0.022}$	$\underline{0.876 \pm 0.041}$	$\underline{0.823 \pm 0.071}$
Isomap	2	0.755 ± 0.040	0.683 ± 0.074	0.721 ± 0.039	0.752 ± 0.040	0.794 ± 0.054	0.755 ± 0.040	0.735 ± 0.066
Opt. App. Eigenmaps (Ours)	5	$\underline{0.882 \pm 0.034}$	$\mathbf{0.874 \pm 0.055}$	$\underline{0.858 \pm 0.046}$	$\underline{0.880 \pm 0.034}$	0.908 ± 0.026	$\underline{0.882 \pm 0.034}$	$\mathbf{0.896 \pm 0.044}$
Laplacian Eigenmaps	5	0.820 ± 0.058	0.707 ± 0.141	0.809 ± 0.053	0.818 ± 0.058	0.833 ± 0.074	0.820 ± 0.058	0.757 ± 0.119
PCA	5	0.818 ± 0.040	0.746 ± 0.080	0.774 ± 0.050	0.816 ± 0.040	0.869 ± 0.039	0.818 ± 0.040	0.792 ± 0.064
UMAP	5	$\mathbf{0.898 \pm 0.043}$	0.830 ± 0.094	$\mathbf{0.859 \pm 0.066}$	$\mathbf{0.897 \pm 0.043}$	$\mathbf{0.942 \pm 0.020}$	$\mathbf{0.898 \pm 0.043}$	0.862 ± 0.073
t-SNE	5	0.875 ± 0.042	0.795 ± 0.093	0.832 ± 0.065	0.874 ± 0.043	$\underline{0.925 \pm 0.022}$	0.875 ± 0.042	0.834 ± 0.072
Isomap	5	0.876 ± 0.030	$\underline{0.844 \pm 0.064}$	0.842 ± 0.046	0.874 ± 0.030	0.914 ± 0.027	0.876 ± 0.030	$\underline{0.872 \pm 0.052}$
Opt. App. Eigenmaps (Ours)	10	$\mathbf{0.915 \pm 0.028}$	$\mathbf{0.917 \pm 0.045}$	$\underline{0.895 \pm 0.042}$	$\mathbf{0.914 \pm 0.029}$	$\underline{0.937 \pm 0.017}$	$\mathbf{0.915 \pm 0.028}$	$\mathbf{0.932 \pm 0.036}$
Laplacian Eigenmaps	10	0.819 ± 0.167	0.707 ± 0.193	$\mathbf{0.899 \pm 0.141}$	0.817 ± 0.168	0.761 ± 0.174	0.819 ± 0.167	0.790 ± 0.108
PCA	10	0.855 ± 0.034	0.780 ± 0.077	0.811 ± 0.051	0.854 ± 0.034	0.907 ± 0.025	0.855 ± 0.034	0.821 ± 0.060
UMAP	10	$\underline{0.899 \pm 0.041}$	0.830 ± 0.091	0.860 ± 0.064	$\underline{0.898 \pm 0.042}$	$\mathbf{0.943 \pm 0.018}$	$\underline{0.899 \pm 0.041}$	0.863 ± 0.071
t-SNE	10	0.876 ± 0.039	0.793 ± 0.084	0.832 ± 0.060	0.874 ± 0.040	0.926 ± 0.022	0.876 ± 0.039	0.832 ± 0.064
Isomap	10	0.895 ± 0.038	$\underline{0.852 \pm 0.080}$	0.860 ± 0.059	0.894 ± 0.038	0.934 ± 0.020	0.895 ± 0.038	$\underline{0.880 \pm 0.063}$

Table 10. Average manifold learning results on CALTECH256 with DINOv2 embeddings. The mean and standard deviation over 32 runs is provided. The best and second best results are in bold and underlined respectively.

CALTECH256 - DINOv2								
Method	k	NMI	ARI	Comp.	AMI	Homo.	V-Meas.	FMI
Opt. App. Eigenmaps (Ours)	2	0.611 ± 0.027	0.078 ± 0.026	0.594 ± 0.030	0.196 ± 0.053	0.629 ± 0.024	0.611 ± 0.027	0.092 ± 0.031
Laplacian Eigenmaps	2	0.650 ± 0.036	0.121 ± 0.037	0.645 ± 0.038	0.311 ± 0.061	0.655 ± 0.036	0.650 ± 0.036	0.134 ± 0.036
PCA	2	0.603 ± 0.029	0.063 ± 0.020	0.583 ± 0.032	0.165 ± 0.050	0.626 ± 0.027	0.603 ± 0.029	0.077 ± 0.026
UMAP	2	$\underline{0.834 \pm 0.014}$	$\mathbf{0.448 \pm 0.084}$	$\mathbf{0.810 \pm 0.020}$	$\underline{0.654 \pm 0.048}$	$\underline{0.860 \pm 0.013}$	$\underline{0.834 \pm 0.014}$	$\mathbf{0.476 \pm 0.073}$
t-SNE	2	$\mathbf{0.840 \pm 0.015}$	$\underline{0.419 \pm 0.071}$	$\underline{0.809 \pm 0.021}$	$\mathbf{0.660 \pm 0.036}$	$\mathbf{0.873 \pm 0.011}$	$\mathbf{0.840 \pm 0.015}$	$\underline{0.457 \pm 0.059}$
Isomap	2	0.655 ± 0.021	0.138 ± 0.045	0.633 ± 0.024	0.273 ± 0.063	0.678 ± 0.019	0.655 ± 0.021	0.155 ± 0.051
Opt. App. Eigenmaps (Ours)	5	0.714 ± 0.020	0.246 ± 0.062	0.696 ± 0.024	0.411 ± 0.058	0.733 ± 0.019	0.714 ± 0.020	0.263 ± 0.067
Laplacian Eigenmaps	5	0.724 ± 0.024	0.209 ± 0.044	0.712 ± 0.026	0.445 ± 0.053	0.737 ± 0.026	0.724 ± 0.024	0.223 ± 0.042
PCA	5	0.696 ± 0.019	0.184 ± 0.039	0.671 ± 0.023	0.355 ± 0.054	0.722 ± 0.015	0.696 ± 0.019	0.206 ± 0.043
UMAP	5	$\mathbf{0.843 \pm 0.013}$	$\mathbf{0.465 \pm 0.081}$	$\mathbf{0.820 \pm 0.020}$	$\mathbf{0.675 \pm 0.039}$	$\mathbf{0.868 \pm 0.011}$	$\mathbf{0.843 \pm 0.013}$	$\mathbf{0.493 \pm 0.068}$
t-SNE	5	$\underline{0.832 \pm 0.015}$	$\underline{0.394 \pm 0.066}$	$\underline{0.800 \pm 0.020}$	$\underline{0.642 \pm 0.039}$	$\underline{0.866 \pm 0.011}$	$\underline{0.832 \pm 0.015}$	$\underline{0.432 \pm 0.056}$
Isomap	5	0.754 ± 0.017	0.319 ± 0.067	0.732 ± 0.021	0.486 ± 0.056	0.778 ± 0.015	0.754 ± 0.017	0.341 ± 0.067
Opt. App. Eigenmaps (Ours)	10	0.759 ± 0.016	0.322 ± 0.060	0.743 ± 0.021	0.508 ± 0.047	0.775 ± 0.014	0.759 ± 0.016	0.339 ± 0.062
Laplacian Eigenmaps	10	0.758 ± 0.016	0.257 ± 0.042	0.743 ± 0.021	0.508 ± 0.045	0.774 ± 0.015	0.758 ± 0.016	0.272 ± 0.039
PCA	10	0.747 ± 0.015	0.267 ± 0.049	0.721 ± 0.020	0.466 ± 0.052	0.776 ± 0.013	0.747 ± 0.015	0.293 ± 0.050
UMAP	10	$\mathbf{0.843 \pm 0.013}$	$\mathbf{0.462 \pm 0.083}$	$\mathbf{0.819 \pm 0.021}$	$\mathbf{0.673 \pm 0.040}$	$\mathbf{0.868 \pm 0.011}$	$\mathbf{0.843 \pm 0.013}$	$\mathbf{0.490 \pm 0.070}$
t-SNE	10	$\underline{0.823 \pm 0.015}$	0.379 ± 0.066	$\underline{0.792 \pm 0.021}$	$\underline{0.623 \pm 0.039}$	$\underline{0.857 \pm 0.011}$	$\underline{0.823 \pm 0.015}$	0.416 ± 0.055
Isomap	10	0.799 ± 0.015	$\underline{0.409 \pm 0.072}$	0.777 ± 0.019	0.582 ± 0.050	0.823 ± 0.012	0.799 ± 0.015	$\underline{0.433 \pm 0.068}$

Table 11. Average manifold learning results on CALTECH256 with CLIP embeddings. The mean and standard deviation over 32 runs is provided. The best and second best results are in bold and underlined respectively.

CALTECH256 - CLIP								
Method	k	NMI	ARI	Comp.	AMI	Homo.	V-Meas.	FMI
Opt. App. Eigenmaps (Ours)	2	0.596 ± 0.031	0.069 ± 0.030	0.578 ± 0.033	0.159 ± 0.050	0.615 ± 0.029	0.596 ± 0.031	0.083 ± 0.036
Laplacian Eigenmaps	2	0.600 ± 0.055	0.075 ± 0.032	0.599 ± 0.055	0.225 ± 0.076	0.602 ± 0.057	0.600 ± 0.055	0.088 ± 0.032
PCA	2	0.599 ± 0.031	0.062 ± 0.022	0.577 ± 0.034	0.153 ± 0.046	0.622 ± 0.028	0.599 ± 0.031	0.076 ± 0.028
UMAP	2	$\underline{0.844 \pm 0.015}$	$\mathbf{0.472 \pm 0.094}$	$\underline{0.819 \pm 0.021}$	$\underline{0.673 \pm 0.051}$	$\underline{0.869 \pm 0.013}$	$\underline{0.844 \pm 0.015}$	$\underline{0.501 \pm 0.081}$
t-SNE	2	$\mathbf{0.862 \pm 0.011}$	$\underline{0.465 \pm 0.079}$	$\mathbf{0.832 \pm 0.019}$	$\mathbf{0.709 \pm 0.031}$	$\mathbf{0.895 \pm 0.007}$	$\mathbf{0.862 \pm 0.011}$	$\mathbf{0.503 \pm 0.064}$
Isomap	2	0.643 ± 0.020	0.115 ± 0.035	0.622 ± 0.023	0.249 ± 0.065	0.666 ± 0.018	0.643 ± 0.020	0.132 ± 0.041
Opt. App. Eigenmaps (Ours)	5	0.685 ± 0.022	0.226 ± 0.068	0.666 ± 0.026	0.348 ± 0.055	0.704 ± 0.020	0.685 ± 0.022	0.243 ± 0.073
Laplacian Eigenmaps	5	0.688 ± 0.024	0.160 ± 0.029	0.685 ± 0.026	0.390 ± 0.051	0.692 ± 0.025	0.688 ± 0.024	0.173 ± 0.029
PCA	5	0.677 ± 0.020	0.165 ± 0.044	0.653 ± 0.022	0.315 ± 0.064	0.704 ± 0.019	0.677 ± 0.020	0.186 ± 0.052
UMAP	5	$\mathbf{0.857 \pm 0.013}$	$\mathbf{0.501 \pm 0.089}$	$\mathbf{0.835 \pm 0.020}$	$\mathbf{0.704 \pm 0.040}$	$\underline{0.881 \pm 0.010}$	$\mathbf{0.857 \pm 0.013}$	$\mathbf{0.528 \pm 0.075}$
t-SNE	5	$\underline{0.853 \pm 0.011}$	$\underline{0.430 \pm 0.070}$	$\underline{0.821 \pm 0.019}$	$\underline{0.686 \pm 0.033}$	$\mathbf{0.887 \pm 0.007}$	$\underline{0.853 \pm 0.011}$	$\underline{0.469 \pm 0.056}$
Isomap	5	0.742 ± 0.017	0.291 ± 0.056	0.721 ± 0.021	0.463 ± 0.059	0.765 ± 0.015	0.742 ± 0.017	0.313 ± 0.056
Opt. App. Eigenmaps (Ours)	10	0.734 ± 0.018	0.309 ± 0.071	0.718 ± 0.022	0.456 ± 0.050	0.750 ± 0.015	0.734 ± 0.018	0.325 ± 0.073
Laplacian Eigenmaps	10	0.735 ± 0.020	0.223 ± 0.038	0.728 ± 0.020	0.475 ± 0.048	0.742 ± 0.025	0.735 ± 0.020	0.236 ± 0.038
PCA	10	0.724 ± 0.014	0.237 ± 0.051	0.698 ± 0.018	0.414 ± 0.062	0.752 ± 0.013	0.724 ± 0.014	0.262 ± 0.056
UMAP	10	$\mathbf{0.856 \pm 0.014}$	$\mathbf{0.494 \pm 0.088}$	$\mathbf{0.833 \pm 0.021}$	$\mathbf{0.702 \pm 0.040}$	$\mathbf{0.881 \pm 0.011}$	$\mathbf{0.856 \pm 0.014}$	$\mathbf{0.522 \pm 0.074}$
t-SNE	10	$\underline{0.844 \pm 0.012}$	$\underline{0.417 \pm 0.070}$	$\underline{0.812 \pm 0.019}$	$\underline{0.667 \pm 0.036}$	$\underline{0.879 \pm 0.008}$	$\underline{0.844 \pm 0.012}$	$\underline{0.456 \pm 0.057}$
Isomap	10	0.788 ± 0.013	0.381 ± 0.062	0.767 ± 0.018	0.560 ± 0.054	0.810 ± 0.012	0.788 ± 0.013	0.403 ± 0.059

Table 12. Average manifold learning results on CIFAR100 with DINOv2 embeddings. The mean and standard deviation over 32 runs is provided. The best and second best results are in bold and underlined respectively.

CIFAR100 - DINOv2								
Method	k	NMI	ARI	Comp.	AMI	Homo.	V-Meas.	FMI
Opt. App. Eigenmaps (Ours)	2	0.568 ± 0.016	0.164 ± 0.036	0.576 ± 0.016	0.412 ± 0.046	0.561 ± 0.020	0.568 ± 0.016	0.191 ± 0.038
Laplacian Eigenmaps	2	0.590 ± 0.032	0.186 ± 0.037	0.577 ± 0.032	0.426 ± 0.050	0.603 ± 0.034	0.590 ± 0.032	0.206 ± 0.038
PCA	2	0.492 ± 0.017	0.098 ± 0.019	0.472 ± 0.020	0.273 ± 0.046	0.515 ± 0.015	0.492 ± 0.017	0.118 ± 0.024
UMAP	2	$\mathbf{0.745 \pm 0.016}$	$\mathbf{0.433 \pm 0.040}$	$\mathbf{0.716 \pm 0.018}$	$\mathbf{0.635 \pm 0.040}$	$\mathbf{0.776 \pm 0.021}$	$\mathbf{0.745 \pm 0.016}$	$\mathbf{0.457 \pm 0.043}$
t-SNE	2	$\underline{0.740 \pm 0.014}$	$\underline{0.412 \pm 0.038}$	$\underline{0.708 \pm 0.016}$	$\underline{0.626 \pm 0.038}$	$\underline{0.775 \pm 0.018}$	$\underline{0.740 \pm 0.014}$	$\underline{0.439 \pm 0.041}$
Isomap	2	0.533 ± 0.018	0.142 ± 0.032	0.512 ± 0.020	0.332 ± 0.051	0.556 ± 0.018	0.533 ± 0.018	0.163 ± 0.036
Opt. App. Eigenmaps (Ours)	5	0.583 ± 0.017	0.176 ± 0.029	0.602 ± 0.023	0.442 ± 0.034	0.565 ± 0.016	0.583 ± 0.017	0.208 ± 0.030
Laplacian Eigenmaps	5	0.640 ± 0.019	0.239 ± 0.036	0.622 ± 0.018	0.492 ± 0.047	0.659 ± 0.024	0.640 ± 0.019	0.258 ± 0.039
PCA	5	0.601 ± 0.013	0.206 ± 0.031	0.576 ± 0.015	0.428 ± 0.049	0.629 ± 0.015	0.601 ± 0.013	0.228 ± 0.036
UMAP	5	$\mathbf{0.750 \pm 0.016}$	$\mathbf{0.447 \pm 0.041}$	$\mathbf{0.722 \pm 0.018}$	$\mathbf{0.644 \pm 0.039}$	$\mathbf{0.781 \pm 0.020}$	$\mathbf{0.750 \pm 0.016}$	$\mathbf{0.470 \pm 0.043}$
t-SNE	5	$\underline{0.736 \pm 0.015}$	$\underline{0.403 \pm 0.036}$	$\underline{0.705 \pm 0.017}$	$\underline{0.621 \pm 0.037}$	$\underline{0.771 \pm 0.018}$	$\underline{0.736 \pm 0.015}$	$\underline{0.430 \pm 0.039}$
Isomap	5	0.647 ± 0.017	0.291 ± 0.048	0.622 ± 0.018	0.495 ± 0.049	0.673 ± 0.020	0.647 ± 0.017	0.313 ± 0.052
Opt. App. Eigenmaps (Ours)	10	0.613 ± 0.013	0.214 ± 0.039	0.622 ± 0.016	0.477 ± 0.036	0.605 ± 0.017	0.613 ± 0.013	0.238 ± 0.040
Laplacian Eigenmaps	10	0.669 ± 0.018	0.273 ± 0.038	0.649 ± 0.017	0.531 ± 0.048	0.690 ± 0.024	0.669 ± 0.018	0.292 ± 0.041
PCA	10	0.649 ± 0.018	0.268 ± 0.042	0.622 ± 0.017	0.495 ± 0.054	0.679 ± 0.023	0.649 ± 0.018	0.292 ± 0.047
UMAP	10	$\mathbf{0.751 \pm 0.015}$	$\mathbf{0.448 \pm 0.038}$	$\mathbf{0.722 \pm 0.017}$	$\mathbf{0.644 \pm 0.036}$	$\mathbf{0.781 \pm 0.018}$	$\mathbf{0.751 \pm 0.015}$	$\mathbf{0.472 \pm 0.040}$
t-SNE	10	$\underline{0.734 \pm 0.019}$	$\underline{0.400 \pm 0.044}$	$\underline{0.703 \pm 0.020}$	$\underline{0.617 \pm 0.044}$	$\underline{0.768 \pm 0.023}$	$\underline{0.734 \pm 0.019}$	$\underline{0.426 \pm 0.047}$
Isomap	10	0.689 ± 0.018	0.355 ± 0.055	0.664 ± 0.017	0.556 ± 0.050	0.717 ± 0.023	0.689 ± 0.018	0.376 ± 0.059

Table 13. Average manifold learning results on CIFAR100 with CLIP embeddings. The mean and standard deviation over 32 runs is provided. The best and second best results are in bold and underlined respectively.

CIFAR100 - CLIP								
Method	k	NMI	ARI	Comp.	AMI	Homo.	V-Meas.	FMI
Opt. App. Eigenmaps (Ours)	2	0.499 ± 0.021	0.111 ± 0.024	0.488 ± 0.024	0.300 ± 0.040	0.511 ± 0.020	0.499 ± 0.021	0.132 ± 0.027
Laplacian Eigenmaps	2	0.392 ± 0.064	0.055 ± 0.028	0.414 ± 0.068	0.208 ± 0.080	0.373 ± 0.063	0.392 ± 0.064	0.088 ± 0.035
PCA	2	0.423 ± 0.028	0.052 ± 0.012	0.406 ± 0.031	0.177 ± 0.036	0.442 ± 0.025	0.423 ± 0.028	0.072 ± 0.016
UMAP	2	$\underline{0.656 \pm 0.022}$	$\mathbf{0.320 \pm 0.045}$	$\underline{0.628 \pm 0.024}$	$\underline{0.506 \pm 0.049}$	$\underline{0.686 \pm 0.023}$	$\underline{0.656 \pm 0.022}$	$\underline{0.345 \pm 0.048}$
t-SNE	2	$\mathbf{0.668 \pm 0.021}$	$\underline{0.320 \pm 0.038}$	$\mathbf{0.639 \pm 0.024}$	$\mathbf{0.523 \pm 0.044}$	$\mathbf{0.701 \pm 0.021}$	$\mathbf{0.668 \pm 0.021}$	$\mathbf{0.346 \pm 0.041}$
Isomap	2	0.464 ± 0.019	0.083 ± 0.019	0.448 ± 0.022	0.240 ± 0.044	0.481 ± 0.017	0.464 ± 0.019	0.103 ± 0.024
Opt. App. Eigenmaps (Ours)	5	0.555 ± 0.019	0.159 ± 0.027	0.552 ± 0.022	0.388 ± 0.040	0.557 ± 0.020	0.555 ± 0.019	0.180 ± 0.030
Laplacian Eigenmaps	5	0.487 ± 0.051	0.101 ± 0.031	0.520 ± 0.047	0.333 ± 0.059	0.459 ± 0.056	0.487 ± 0.051	0.140 ± 0.032
PCA	5	0.504 ± 0.020	0.116 ± 0.026	0.481 ± 0.023	0.286 ± 0.048	0.528 ± 0.019	0.504 ± 0.020	0.138 ± 0.030
UMAP	5	$\mathbf{0.667 \pm 0.021}$	$\mathbf{0.342 \pm 0.041}$	$\mathbf{0.641 \pm 0.023}$	$\mathbf{0.525 \pm 0.044}$	$\underline{0.696 \pm 0.021}$	$\mathbf{0.667 \pm 0.021}$	$\mathbf{0.365 \pm 0.045}$
t-SNE	5	$\underline{0.665 \pm 0.021}$	$\underline{0.310 \pm 0.039}$	$\underline{0.636 \pm 0.023}$	$\underline{0.519 \pm 0.044}$	$\mathbf{0.698 \pm 0.021}$	$\underline{0.665 \pm 0.021}$	$\underline{0.336 \pm 0.042}$
Isomap	5	0.561 ± 0.019	0.176 ± 0.036	0.541 ± 0.021	0.375 ± 0.054	0.582 ± 0.020	0.561 ± 0.019	0.197 ± 0.041
Opt. App. Eigenmaps (Ours)	10	0.566 ± 0.021	0.173 ± 0.037	0.577 ± 0.023	0.415 ± 0.042	0.555 ± 0.023	0.566 ± 0.021	0.200 ± 0.037
Laplacian Eigenmaps	10	0.551 ± 0.031	0.137 ± 0.037	0.579 ± 0.025	0.407 ± 0.049	0.526 ± 0.039	0.551 ± 0.031	0.174 ± 0.036
PCA	10	0.557 ± 0.018	0.172 ± 0.035	0.532 ± 0.021	0.363 ± 0.047	0.584 ± 0.017	0.557 ± 0.018	0.195 ± 0.039
UMAP	10	$\mathbf{0.667 \pm 0.021}$	$\mathbf{0.343 \pm 0.045}$	$\mathbf{0.641 \pm 0.024}$	$\mathbf{0.525 \pm 0.044}$	$\mathbf{0.696 \pm 0.021}$	$\mathbf{0.667 \pm 0.021}$	$\mathbf{0.366 \pm 0.048}$
t-SNE	10	$\underline{0.653 \pm 0.027}$	$\underline{0.292 \pm 0.049}$	$\underline{0.625 \pm 0.028}$	$\underline{0.501 \pm 0.056}$	$\underline{0.685 \pm 0.028}$	$\underline{0.653 \pm 0.027}$	$\underline{0.318 \pm 0.053}$
Isomap	10	0.588 ± 0.019	0.211 ± 0.039	0.567 ± 0.021	0.414 ± 0.051	0.611 ± 0.021	0.588 ± 0.019	0.232 ± 0.044

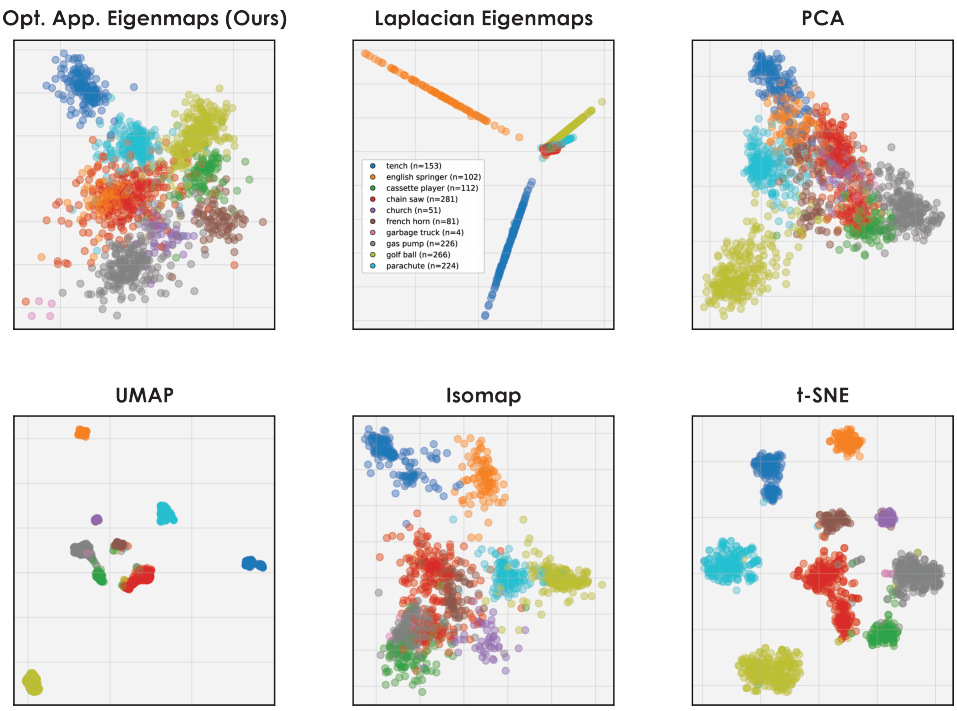

Figure 23. 2D visualization of a random subset of the Imagenette dataset using CLIP embeddings. Points are colored by ground truth class labels. Our learned spectral basis (Optimal Approximation Eigenmaps) produces meaningful clusters that separate the image manifold according to semantic content, with less overlap than in PCA and Isomap, while accurately preserving the smooth manifold structure without artificially overclustering it as in t-SNE, Umap, and Laplacian Eigenmaps.

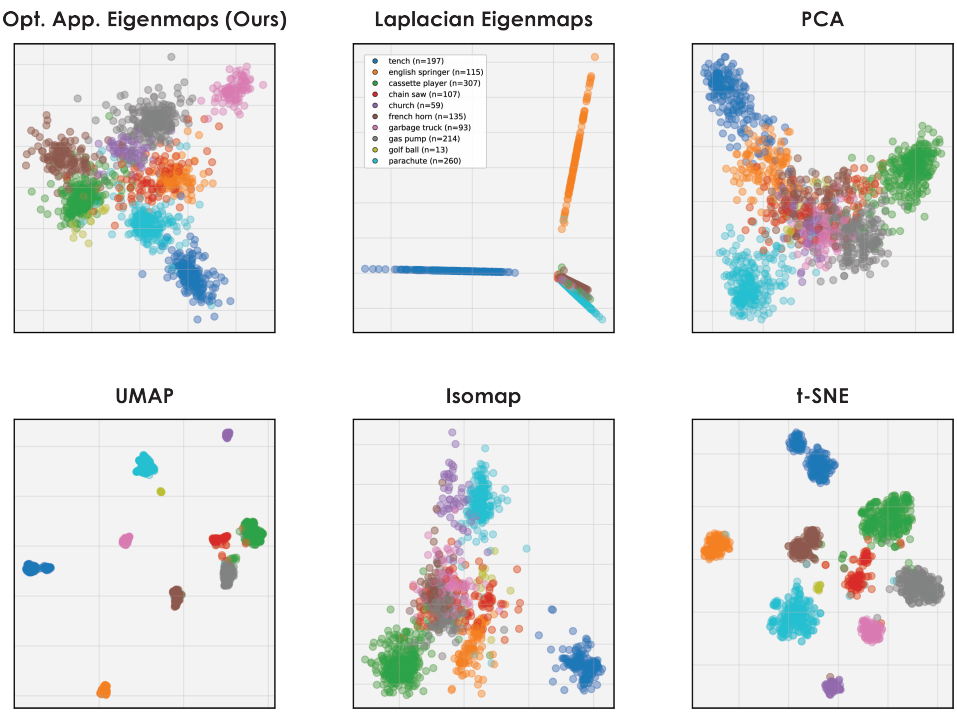

Figure 24. 2D visualization of another random subset of the Imagenette dataset using CLIP embeddings.

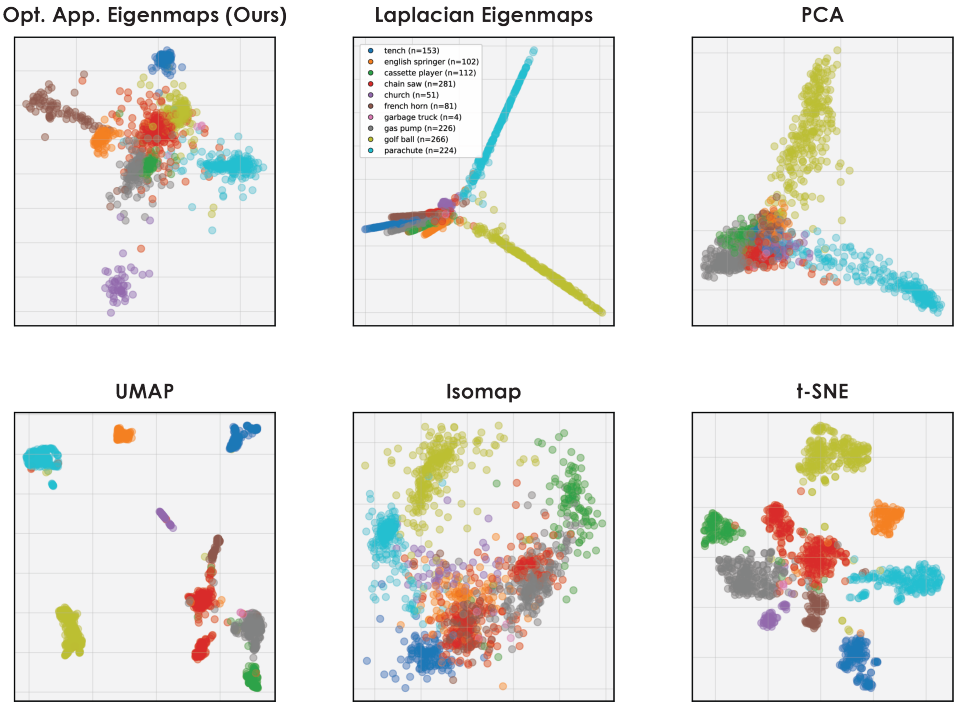

Figure 25. 2D visualization of a random subset of the Imagenette dataset using DINOv2 embeddings.

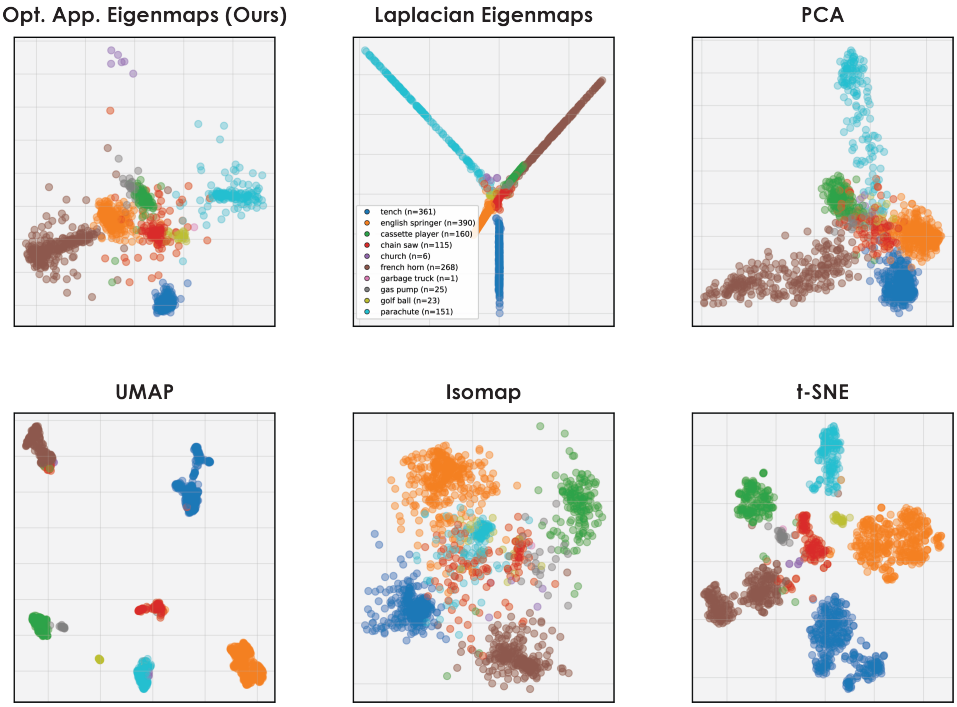

Figure 26. 2D visualization of another random subset of the Imagenette dataset using DINOv2 embeddings.

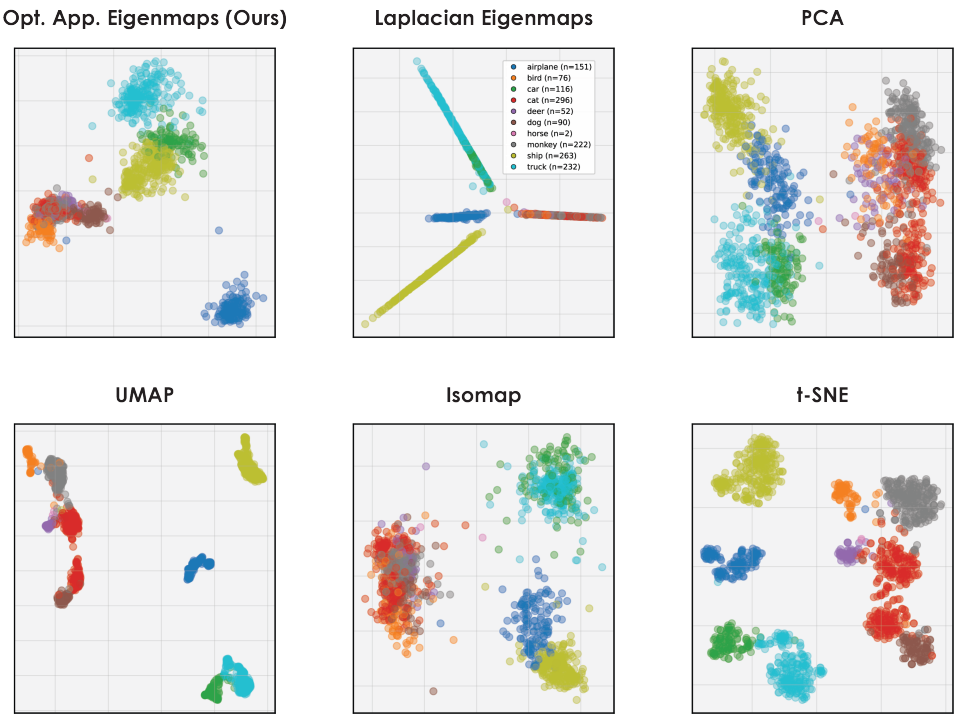

Figure 27. 2D visualization of a random subset of the STL-10 dataset using CLIP embeddings.

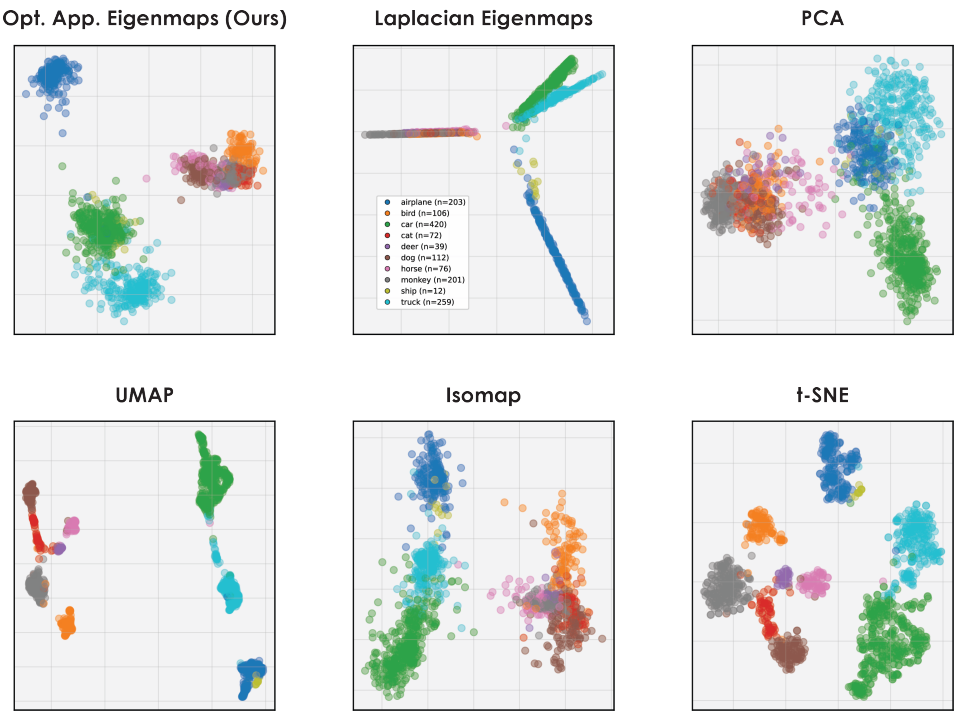

Figure 28. 2D visualization of another random subset of the STL-10 dataset using CLIP embeddings.

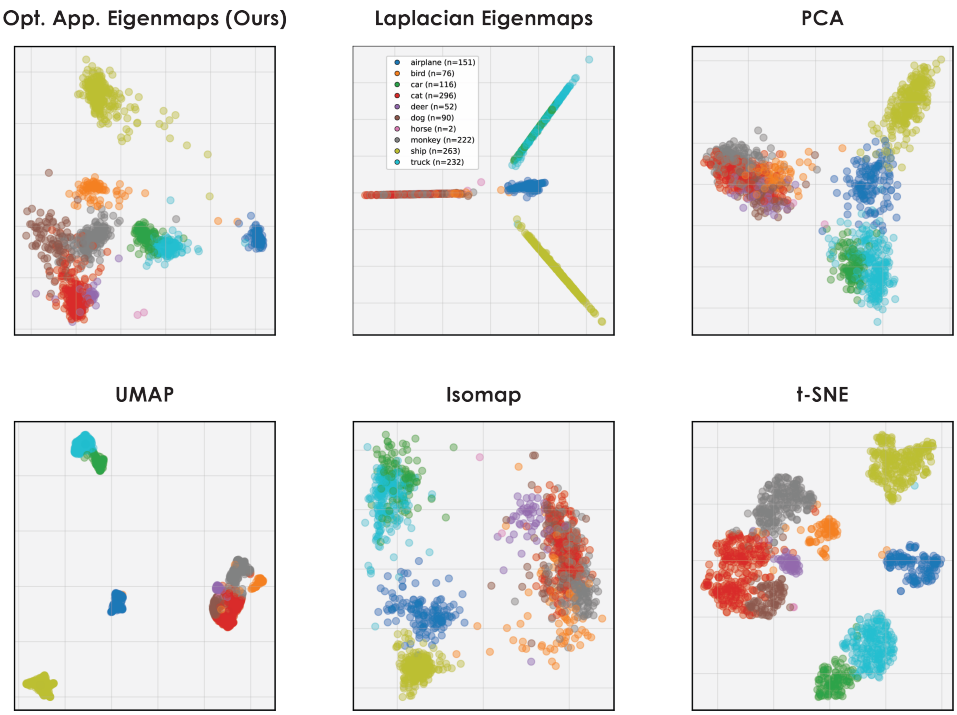

Figure 29. 2D visualization of a random subset of the STL-10 dataset using DINOv2 embeddings.

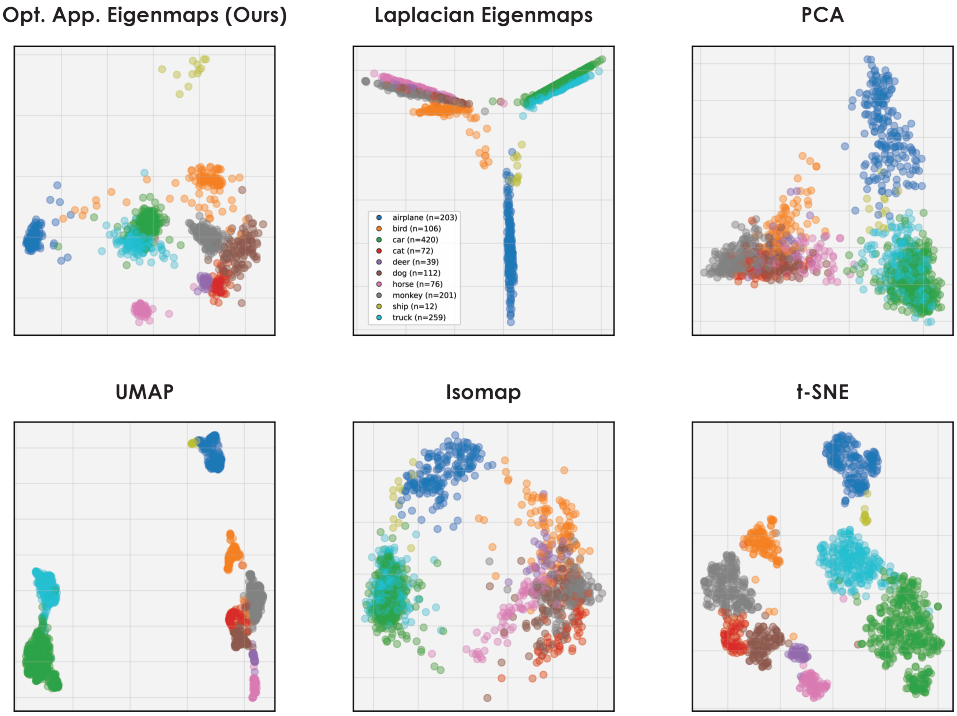

Figure 30. 2D visualization of another random subset of the STL-10 dataset using DINOv2 embeddings.