GNN-Based Inductive Dimensionality Reduction on UMAP Graphs

F. Grötschla[1] and S. van Wageningen[2] and A. C. Telea[2]

[1]ETH Zurich, Zürich, Switzerland
[2] Department of Information and Computing Science, Utrecht University, Utrecht, Netherlands

Abstract

Dimensionality reduction methods such as UMAP and t-SNE produce high-quality 2D embeddings but are transductive: they must be re-optimized for every new dataset. We observe that UMAP naturally decomposes into graph construction and layout optimization, and replace the layout step with a graph neural network trained on UMAP's cross-entropy objective. As the GNN operates on graph topology rather than raw data, a single trained model generalizes across domains: On four held-out datasets,our method approaches UMAP's local neighborhood preservation while achieving lower pairwise distance stress, without any per-dataset retraining.

CCS Concepts

• ***Mathematics of computing*** → ***Dimensionality reduction;*** • ***Computer systems organization*** → ***Neural networks;*** • ***Human-centered computing*** → *Graph drawings;*

1. Introduction

Dimensionality Reduction (DR) is fundamental for exploratory visualization of high-dimensional data. Neighbor-embedding techniques such as t-SNE [MH08] and UMAP [MHSG18] are frequently used for DR. Yet, such methods are *transductive*: they optimize an embedding for a given dataset and must re-run from scratch when data changes. This limits their use in interactive or batch-processing settings where embeddings must be computed repeatedly, e.g. for tools that combine CLIP embeddings with UMAP projections [GLCW24]. Existing parametric extensions [VDM09,SMG21] train neural networks to approximate these embeddings, but still require a full training-run per dataset and do not generalize across domains – *e.g.*, a model trained to project image data cannot be used for text.

UMAP naturally decomposes into (1) building a weighted k-nearest neighbor graph using fuzzy simplicial sets; and (2) layout optimization, where low-dimensional coordinates are found by minimizing a cross-entropy objective over the graph. This decomposition bridges DR and graph drawing [**?**]: (1) produces a weighted graph, and (2) solves a graph layout problem. We exploit this by replacing UMAP's iterative layout optimizer with a GNN adapted from CoRe-GD [GMVW24], a recent architecture for scalable graph drawing (see Fig. 1). Our model operates on graph topology rather than raw data so it naturally transfers across datasets: once trained on small graphs from several source domains, it embeds *unseen* data from *different domains* in a single forward pass. Our contributions are:

- a GNN architecture for inductive DR that operates on UMAP-weighted k-NN graphs with negative-sampling rewiring;
- a training procedure combining UMAP's cross-entropy loss with a replay buffer for deep unrolling;
- evaluation on 7 datasets (4 held-out) shows competitive embedding quality with no per-dataset retraining.

All code is publicly available at `https://github.com/floriangroetschla/gnn-dr`.

2. Related Work

Dimensionality reduction: t-SNE [MH08] and UMAP [MHSG18], the main neighbor-embedding methods for DR, build a neighborhood graph and optimize low-dimensional coordinates to preserve its structure using different similarity kernels and costs. Parametric t-SNE [VDM09] and Parametric UMAP [SMG21] train neural networks to predict embeddings, enabling out-of-sample (OOS) extension but still need a full training run per dataset and do not transfer to new domains. NNP [EHST20] learns to mimic any projection, *e.g.* t-SNE and UMAP, on a data subset with OOS extension to the full dataset but also needs retraining when datasets change. h-NNE [SKSS22] avoids iterative optimization via hierarchical nearest-neighbor graphs achieving fast runtimes with OOS support.
Graph drawing: Several recent methods use GNNs to layout optimization: Giovannangeli et al. [GLA*22] train GCNs on stress-based objectives; Both et al. [BDYB23] accelerate force-directed layouts 10× to 100×; and CoRe-GD [GMVW24] introduces hierarchical coarsening with Gate Recurrent Unit (GRU)-based message passing. NNP-NET [HMvW*25] adapts NNP [EHST20] to lay out large graphs via t-SNE projection of the graph distance matrix. All

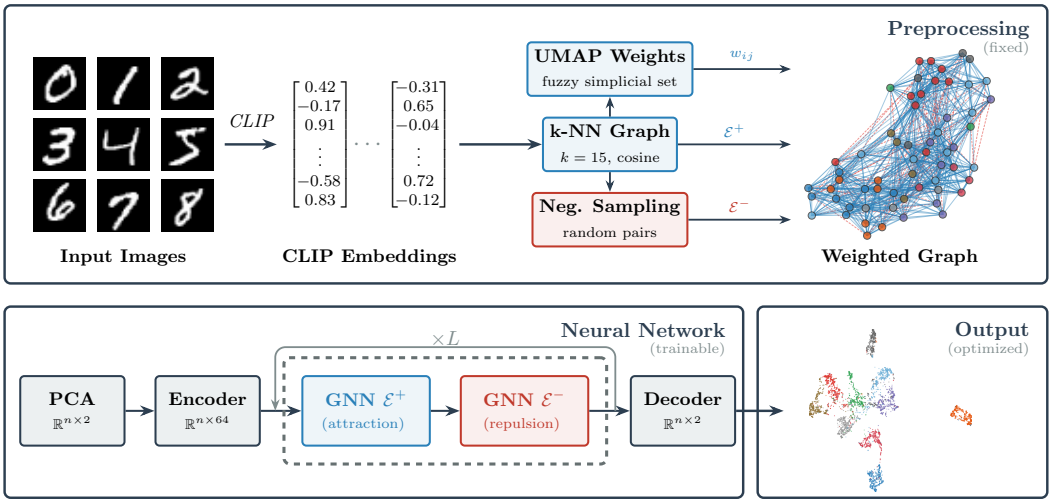

Figure 1: *Pipeline overview. CLIP embeddings are used to construct a k-NN graph with UMAP weights w_{ij}. PCA-initialized positions are iteratively refined by a GRU-based GNN alternating between structural convolutions (attraction) and negative-edge convolutions (repulsion).*

these methods operate on *explicitly* given graphs. Our work connects these fields by applying graph-drawing machinery to UMAP-weighted k-NN graphs, yielding a single model that generalizes across domains.

3. Method

Given high-dimensional data, we construct a k-NN graph with UMAP fuzzy simplicial set weights (§3.1), initialize node positions via PCA, and iteratively refine them with a GRU-based GNN (§3.2) trained on UMAP's cross-entropy objective (§3.3).

3.1. Graph Construction and Input Features

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^D$ be n samples (D=512 CLIP [RKH*21] embeddings in our experiments). We construct a k-NN graph $G = (V, \mathcal{E}^+, \mathcal{E}^-, w)$ for each sample in $\mathcal{X}$, where $\mathcal{E}^+$ are k-NN edges and $\mathcal{E}^-$ negative pairs that are randomly sampled (we use k=15 and cosine distance in all experiments, see §4). Edge weights $w_{ij} \in (0,1]$ are computed via UMAP's fuzzy simplicial set construction [MHSG18], which assigns directed membership strengths based on locally adaptive bandwidths and symmetrizes them via fuzzy union. Weights serve as both features for edges $e \in \mathcal{E}^+$ during message passing and supervision signal in the loss. Features of nodes $v \in V$ are initialized with 2D coordinates by doing PCA on $\mathcal{X}$ providing a coarse layout while keeping our model independent on the data dimensionality D. While we use UMAP's fuzzy simplicial set construction here, the architecture operates on any weighted k-NN graph. Alternative weighting schemes (*e.g.*, heat-kernel weights) could be substituted without architectural changes.

3.2. Architecture

Our model adapts the GNN-based encode/process/decode architecture of CoRe-GD [GMVW24] with iterative refinement (Fig. 1). An MLP encoder maps 2D PCA coordinates to hidden states $\mathbf{h}_i^{(0)} \in \mathbb{R}^H$ (H=64); here and next, superscripts denote iteration numbers. An MLP decoder outputs 2D positions $\hat{\mathbf{y}}_i = \mathrm{MLP}_{\mathrm{dec}}(\mathbf{h}_i^{(T)})$ after the final iteration. The model performs L iterations, each with two phases.

Phase 1: Structural convolutions. Two GRU-based edge convolutions operate sequentially on k-NN edges. At iteration t, each message is computed by an MLP that takes source and target hidden states together with the UMAP edge weight, allowing the network to modulate messages between nodes j and i according to w_{ij}:

$$\mathbf{m}_{j\to i} = \mathrm{MLP}_{\mathrm{edge}}\left([\mathbf{h}_j^{(t)} \,\|\, \mathbf{h}_i^{(t)} \,\|\, w_{ij}]\right) \tag{1}$$

Messages are summed and the state (values for iteration $t+1$) is updated using a GRU cell [CvMG*14] to be

$$\mathbf{h}_i^{(t+1)} = \mathrm{GRU}\left(\textstyle\sum_{j\in\mathcal{N}_k(i)} \mathbf{m}_{j\to i},\ \mathbf{h}_i^{(t)}\right), \tag{2}$$

where $\mathcal{N}_k(i)$ are all nodes connected to node i (corresponding to the k-NNs of $\mathbf{x}_i$).

Phase 2: Negative-edge convolutions. A separate GRU convolution processes $10\times|\mathcal{E}^+|$ *negative edges* $\mathcal{E}^-$ (non-neighbor pairs, sampled once during graph construction and fixed over iterations) without edge attributes. Unlike CoRe-GD, which dynamically rewires edges based on intermediate positions, we avoid costly recomputations by using a static negative-edge set. This provides repulsive forces akin to UMAP's repulsion term: without it, the model would only receive attractive signals via k-NN edges and could not push non-neighbors apart. Denoting the state after both convolution phases as $\tilde{\mathbf{h}}_i$, a skip connection $\mathbf{h}_i^{(t+1)} = \mathrm{MLP}_{\mathrm{skip}}([\tilde{\mathbf{h}}_i \,\|\, \tilde{\mathbf{h}}_i + \mathbf{h}_i^{(t)}])$ stabilizes gradient flow across iterations; during training, $L \sim \mathcal{N}(5,1)$ (rounded, ≥ 1) encourages the model to produce intermediate embeddings that can be decoded into the final coordinates at varying depths D.

3.3. Loss and Training

We train with UMAP's fuzzy cross-entropy loss [MHSG18] using the similarity kernel $q_{ij} = (1 + a\|\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j\|^{2b})^{-1}$ with a, b computed

according to UMAP [MHSG18] ($d_{\min}$=0.1, spread=1.0). The loss reads as

$$\mathcal{L} = \underbrace{-\frac{\sum_{(i,j)\in\mathcal{E}^+} w_{ij}\log q_{ij}}{\sum_{(i,j)\in\mathcal{E}^+} w_{ij}}}_{\mathcal{L}_{\text{attract}}} + \gamma \underbrace{\left(-\frac{1}{|\mathcal{E}^-|}\sum_{(i,j')\in\mathcal{E}^-}\log(1-q_{ij'})\right)}_{\mathcal{L}_{\text{repel}}} \tag{3}$$

with γ=5 and 10 negative samples per positive edge (excluding actual neighbors). We create training graphs on-the-fly by subsampling n from a set of sizes with up to 1,500 points from three input datasets $\mathcal{X} \in \{\text{CIFAR-10, MNIST, Fashion-MNIST}\}$. For each subsample, a fresh k-NN graph with UMAP weights, which exposes the model to diverse graph structures and sizes. To enable deep iterative refinement without proportional memory cost, we keep a replay buffer [GMVW24] of capacity 64. Each training step accumulates gradients from one fresh batch (passing through the encoder) and one replayed batch (continuing from stored hidden states, bypassing the encoder). A graph replayed r times undergoes $\sim r \cdot T$ total iterations, effectively deepening unrolling over training. We train for 200 epochs on a single NVIDIA A100 (80 GB), which takes less than 2 days. At inference time, to project unseen data from any domain, we (1) build the k-NN graph with UMAP weights, (2) compute PCA initialization, (3) run the GNN for L=20 iterations; as already mentioned, no model retraining is required.

4. Evaluation

We use CLIP ViT-B/32 [RKH*21] embeddings (D=512) from three training datasets (CIFAR-10, MNIST, Fashion-MNIST) and four held-out datasets (KMNIST, FGVC Aircraft, Oxford Pets, Food-101). Our GNN uses hidden dimension H=64, 2-hidden-layer MLPs with hidden size $4H$=256, LayerNorm, dropout 0.1, sum aggregation, k=15 neighbors, and L=20 iterations at inference. We compare against PCA (our initialization), t-SNE [MH08] (perplexity 30), UMAP [MHSG18] (k=15, $d_{\min}$=0.1), and Parametric UMAP (P-UMAP) [SMG21], the latter being a 3-layer MLP encoder (hidden dimensionality 256) trained on the same three source datasets (30k CLIP embeddings) with the UMAP cross-entropy loss. t-SNE and UMAP are transductive; P-UMAP and our method are both inductive and trained once. We report five projection quality metrics [MBT25]: Trustworthiness [VK06] and Continuity [VK06] measure local neighborhood preservation (false and missing neighbors, respectively); Neighborhood Hit (NH) measures class-label agreement among k-nearest neighbors; Shepard goodness measures rank correlation of pairwise distances; and Scale-Normalized Stress (SNS, ↓) measures pairwise distance distortion. We also report wall-clock execution (inference) time including graph construction. Results for more metrics (14 in total) and more visualizations can be found in the supplementary material.

Embedding Quality. Table 1 shows per-dataset results. Our method closely tracks UMAP on trustworthiness and continuity, with a consistent gap of 1 to 2%. Neighborhood hit is lower for our method, reflecting the cost of generalization: transductive methods can optimize cluster separation *per dataset*, whereas our model uses a single set of weights (obtained from training) for *all datasets*. On scale-normalized stress, our method outperforms UMAP on 5 of 7 datasets, suggesting that iterative message passing preserves more global distance structure than UMAP's stochastic gradient descent. P-UMAP, while trained on the same data and loss, consistently *underperforms* our method on local metrics, indicating that our operating on graph topology provides a stronger inductive bias than learning a direct mapping from features.

Table 1: *Quantitative comparison. Numbers in brackets show the sizes of the projected datsasets.* ***Bold****: best,* underline*: second best per metric per dataset. Time includes k-NN graph construction (all tests done on an AMD EPYC 7742. Our method additionally uses an NVIDIA A6000 GPU.*

		Local			*Global*		
	Method	Tr. ↑	Co. ↑	NH ↑	Sh. ↑	SNS ↓	Time (s)
MNIST (10k)	PCA	0.781	0.931	0.227	**0.638**	**0.146**	0.1
	t-SNE	**0.980**	0.971	**0.923**	0.411	0.154	19
	UMAP	0.959	**0.972**	0.918	0.347	0.218	28
	P-UMAP	0.806	0.950	0.481	0.547	0.150	**0.1**
	Ours	0.954	0.970	0.909	0.478	0.157	6
F-MNIST (10k)	PCA	0.882	0.957	0.453	**0.716**	**0.125**	0.1
	t-SNE	**0.987**	0.979	**0.772**	0.560	0.133	18
	UMAP	0.971	0.983	0.735	0.536	0.226	12
	P-UMAP	0.916	0.973	0.606	0.672	0.144	**0.1**
	Ours	0.966	**0.984**	0.728	0.561	0.161	6
CIFAR-10 (10k)	PCA	0.844	0.942	0.484	**0.699**	0.147	0.1
	t-SNE	**0.977**	0.970	**0.861**	0.562	**0.147**	19
	UMAP	0.958	0.972	0.859	0.637	0.211	12
	P-UMAP	0.904	0.956	0.756	0.641	0.264	**0.1**
	Ours	0.954	**0.975**	0.856	0.613	0.153	6
Held-out (unseen during training):							
KMNIST (10k)	PCA	0.822	0.943	0.174	**0.728**	**0.116**	0.1
	t-SNE	**0.971**	0.945	**0.595**	0.597	0.125	19
	UMAP	0.921	**0.955**	0.485	0.588	0.143	12
	P-UMAP	0.628	0.814	0.136	0.299	0.231	**0.1**
	Ours	0.913	0.954	0.454	0.569	0.164	6
Aircraft (3.3k)	PCA	0.818	0.910	0.085	0.687	0.153	0.1
	t-SNE	**0.949**	0.934	**0.197**	0.657	**0.138**	6
	UMAP	0.926	0.939	0.182	**0.702**	0.266	15
	P-UMAP	0.643	0.770	0.022	0.352	0.219	**0.1**
	Ours	0.928	**0.939**	0.183	0.687	0.163	2
Pets (3.7k)	PCA	0.785	0.885	0.096	**0.577**	0.187	0.1
	t-SNE	**0.959**	0.957	**0.684**	0.518	**0.168**	7
	UMAP	0.945	**0.962**	0.667	0.535	0.321	13
	P-UMAP	0.723	0.835	0.085	0.489	0.271	**0.1**
	Ours	0.946	0.956	0.649	0.465	0.189	2
Food-101 (10k)	PCA	0.727	0.890	0.058	**0.477**	0.173	0.1
	t-SNE	**0.979**	**0.958**	**0.695**	0.389	**0.165**	17
	UMAP	0.964	0.955	0.671	0.355	0.209	12
	P-UMAP	0.551	0.670	0.024	0.092	0.276	**0.1**
	Ours	0.956	0.955	0.646	0.345	0.181	6

Cross-Domain Generalization. Our gap to UMAP on local metrics is similar across training and held-out datasets, even for fine-grained domains (Aircraft, Pets) where class structure differs strongly from training data (CIFAR-10, MNIST, Fashion-MNIST). P-UMAP degrades substantially on held-out data (e.g. trustworthiness drops to 0.55 on Food-101 vs. 0.95 for ours), showing that its MLP encoder overfits to the training domains while our GNN learns transferable layout principles from graph topology. Figure 2 confirms these metric results visually by the created projections.

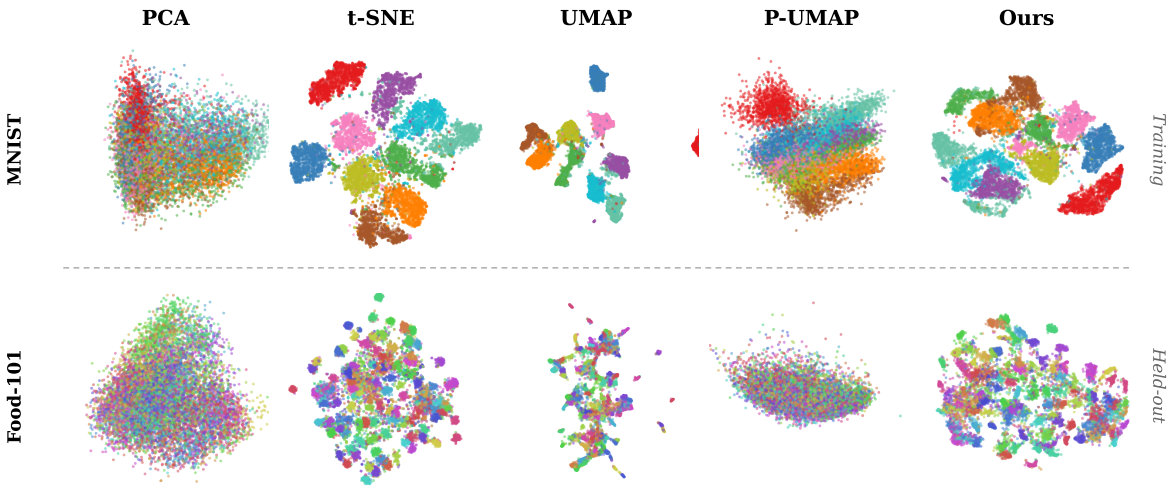

Figure 2: *2D embeddings by PCA, t-SNE, UMAP, P-UMAP, and our method. Top: MNIST (training, 10 classes). Bottom: Food-101 (held-out, 101 classes). Points colored by class. Our method produces cluster structure comparable to UMAP and t-SNE, even on unseen (held-out) domains with many classes. In contrast, P-UMAP degrades on held-out data.*

Runtime. P-UMAP is the fastest inductive method since it only needs a forward pass through a small MLP (see Tab. 1). Our method is slower due to graph construction and PCA preprocessing, which account for ∼80% of its runtime, but remains faster than the transductive baselines. Note that t-SNE and UMAP were run on CPU only; GPU-accelerated versions would narrow the gap.

5. Discussion and Conclusion

We presented an inductive approach to dimensionality reduction that bridges graph drawing and DR by adapting a GNN architecture from graph layout optimization [GMVW24] to UMAP-weighted k-NN graphs. Our model is *amortized*: training cost is paid once, and all subsequent embeddings are produced at inference cost only. Compared to Parametric UMAP, which also amortizes training, our graph-based approach generalizes far better to unseen domains because it operates on topology rather than raw features. Interestingly, our model achieves lower stress than UMAP on most datasets despite not explicitly optimizing for it. A current limitation is that the UMAP hyperparameters $d_{\min}$, spread, and k are fixed at training time: changing them requires retraining the GNN, which shifts the tuning burden from per-dataset optimization to a one-time training cost. Future work includes making these parameters inference-time adjustable, scaling via hierarchical coarsening, extending to other embedding spaces, and progressive refinement for interactive exploration.

References

[BDYB23] Both C., Dehmamy N., Yu R., Barabási A.-L.: Accelerating network layouts using graph neural networks. *Nature Comm 14*, 1 (2023), 1560. 1

[CvMG*14] Cho K., van Merriënboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H., Bengio Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. (EMNLP)* (2014), pp. 1724–1734. doi:10.3115/v1/D14-1179. 2

[EHST20] Espadoto M., Hirata N., Sumiko T., Telea A. C.: Deep learning multidimensional projections. *Inf Vis 19*, 3 (2020), 247–269. 1

[GLA*22] Giovannangeli L., Lalanne F., Auber D., Giot R., Bourqui R.: Toward efficient deep learning for graph drawing (DL4GD). *IEEE TVCG 30*, 2 (2022), 1516–1532. 1

[GLCW24] Grötschla F., Lanzendörfer L. A., Calzavara M., Wattenhofer R.: AEye: A visualization tool for image datasets. In *Proc. IEEE Visualization* (2024), pp. 281–285. 1

[GMVW24] Grötschla F., Mathys J., Veres R., Wattenhofer R.: Core-gd: A hierarchical framework for scalable graph visualization with GNNs. In *Proc. ICLR* (2024). 1, 2, 3, 4

[HMvW*25] Hartskeerl I., Mchedlidze T., van Wageningen S., Vangorp P., Telea A.: NNP-NET: Accelerating t-SNE graph drawing for very large graphs by neural networks. In *Graph Drawing* (2025). 1

[MBT25] Machado A., Behrisch M., Telea A.: Extensible TensorFlow implementations of projection quality metrics. In *Proc. VisGap* (2025). 3

[MH08] Maaten L. v. d., Hinton G.: Visualizing data using t-SNE. *J Mach Learn Res 9*, Nov (2008), 2579–2605. 1, 3

[MHSG18] McInnes L., Healy J., Saul N., Grossberger L.: Umap: Uniform manifold approximation and projection. *Journal of Open Source Software 3*, 29 (2018), 861. doi:10.21105/joss.00861. 1, 2, 3

[RKH*21] Radford A., Kim J. W., Hallacy C., Ramesh A., Goh G., Agarwal S., Sastry G., Askell A., Mishkin P., Clark J., et al.: Learning transferable visual models from natural language supervision. In *Proc. ICML* (2021), pp. 8748–8763. 2, 3

[SKSS22] Sarfraz S., Koulakis M., Seibold C., Stiefelhagen R.: Hierarchical nearest neighbor graph embedding for efficient dimensionality reduction. In *Proc. CVPR* (2022), pp. 336–345. 1

[SMG21] Sainburg T., McInnes L., Gentner T. Q.: Parametric umap embeddings for representation and semisupervised learning. *Neural Comput 33*, 11 (2021), 2881–2907. 1, 3

[VDM09] Van Der Maaten L.: Learning a parametric embedding by preserving local structure. In *Artificial intelligence and statistics* (2009), PMLR, pp. 384–391. 1

[VK06] Venna J., Kaski S.: Local multidimensional scaling. *Neural Networks 19*, 6-7 (2006), 889–899. 3

Appendix A: Full Metric Comparison

Table 2 reports all 14 metrics evaluated, grouped into *local* (neighborhood preservation), *global* (distance/structure preservation), and *error* (distortion measures). The main paper reports five representative metrics; here we include the full set for completeness.

Several patterns emerge from the extended evaluation. On local metrics, t-SNE and UMAP consistently achieve the highest trustworthiness and Jaccard scores, reflecting their per-dataset optimization. Our method ranks second or third on most local metrics, closely trailing UMAP on continuity and class-aware metrics, and clearly ahead of P-UMAP on all held-out datasets. On global metrics, our method achieves the best or second-best scale-normalized stress (SNS) on 6 of 7 datasets, suggesting that iterative GNN message passing preserves global distance structure well. PCA dominates Shepard goodness and Pearson correlation, which is expected since PCA is a linear projection that preserves large-scale variance. On error metrics, our method achieves the lowest ALE on 3 of 7 datasets, indicating low local geometric distortion; MRRE scores are competitive with UMAP despite the model never being trained on these datasets. P-UMAP degrades severely on held-out data across almost all metrics, confirming that its MLP encoder overfits to the training domain features.

Appendix B: Scatter Plot Comparisons

Figure 3 shows 2D projections for all datasets and methods.

Table 2: *Full quantitative comparison across all metrics.* ***Bold****: best,* underline*: second best per metric per dataset. Arrows indicate direction (↑ = higher is better, ↓ = lower is better). Metrics: Trustworthiness (Tr.), Continuity (Co.), Neighborhood Hit (NH), Jaccard index (Jacc.), Class-Aware Trustworthiness (CA-Tr.), Class-Aware Continuity (CA-Co.), Shepard goodness (Sh.), Distance Consistency (DC), Normalized Stress (N. Str.), Scale-Normalized Stress (SNS), Pearson correlation (Pears.), Average Local Error (ALE), MRRE in data space ($MRRE_d$), MRRE in projection space ($MRRE_p$).*

		Local						*Global*					*Error*		
	Method	Tr. ↑	Co. ↑	NH ↑	Jacc. ↑	CA-Tr. ↑	CA-Co. ↑	Sh. ↑	DC ↑	N. Str. ↓	SNS ↓	Pears. ↑	ALE ↓	$MRRE_d$ ↓	$MRRE_p$ ↓
MNIST (10k)	PCA	0.781	0.931	0.227	0.016	0.794	0.939	**0.638**	0.303	**0.300**	**0.146**	**0.655**	**0.096**	0.220	0.060
	t-SNE	**0.980**	0.971	**0.923**	**0.253**	**0.993**	0.987	0.411	0.915	63761	0.154	0.427	0.127	**0.012**	0.021
	UMAP	0.959	**0.972**	0.918	0.154	0.992	**0.990**	0.347	**0.923**	825	0.218	0.337	0.139	0.039	**0.020**
	P-UMAP	0.806	0.950	0.481	0.025	0.841	0.959	0.547	0.521	661263	0.150	0.554	0.111	0.194	0.044
	Ours	0.954	0.970	0.909	0.149	0.987	0.986	0.478	0.760	452	0.157	0.482	0.123	0.044	0.022
F-MNIST (10k)	PCA	0.882	0.957	0.453	0.025	0.908	0.969	**0.716**	0.514	**0.274**	**0.125**	**0.722**	0.159	0.118	0.038
	t-SNE	**0.987**	0.979	**0.772**	**0.237**	**0.995**	0.989	0.560	**0.742**	20451	0.133	0.584	**0.158**	**0.009**	0.016
	UMAP	0.971	0.983	0.735	0.140	0.987	**0.991**	0.536	0.716	498	0.226	0.528	0.197	0.029	0.013
	P-UMAP	0.916	0.973	0.606	0.039	0.943	0.982	0.672	0.657	1055664	0.144	0.680	0.201	0.085	0.024
	Ours	0.966	**0.984**	0.728	0.132	0.984	0.991	0.561	0.737	156	0.161	0.552	0.178	0.033	**0.013**
CIFAR-10 (10k)	PCA	0.844	0.942	0.484	0.016	0.875	0.954	**0.699**	0.560	**0.371**	0.147	**0.697**	0.253	0.156	0.053
	t-SNE	**0.977**	0.970	**0.861**	**0.192**	**0.992**	0.983	0.562	0.854	9740	**0.147**	0.578	**0.218**	**0.015**	0.023
	UMAP	0.958	0.972	0.859	0.105	0.988	0.985	0.637	0.880	176	0.211	0.622	0.229	0.040	0.022
	P-UMAP	0.904	0.956	0.756	0.028	0.950	0.971	0.641	0.787	3539602	0.264	0.615	0.237	0.096	0.041
	Ours	0.954	**0.975**	0.856	0.099	0.985	**0.987**	0.613	**0.887**	71.3	0.153	0.620	0.224	0.045	**0.019**
Held-out (unseen during training):															
KMNIST (10k)	PCA	0.822	0.943	0.174	0.017	0.840	0.964	**0.728**	0.207	**0.251**	**0.116**	**0.741**	**0.106**	0.179	0.051
	t-SNE	**0.971**	0.945	**0.595**	**0.219**	**0.980**	0.977	0.597	0.256	40647	0.125	0.599	0.108	**0.017**	0.042
	UMAP	0.921	**0.955**	0.485	0.114	0.940	**0.983**	0.588	**0.267**	172	0.143	0.582	0.119	0.077	**0.033**
	P-UMAP	0.628	0.814	0.136	0.005	0.663	0.892	0.299	0.190	214123	0.231	0.333	0.177	0.373	0.169
	Ours	0.913	0.954	0.454	0.107	0.931	0.982	0.569	0.238	255	0.164	0.549	0.124	0.086	0.034
Aircraft (3.3k)	PCA	0.818	0.910	0.085	0.038	0.823	0.990	0.687	0.135	**0.368**	0.153	0.683	0.281	0.184	0.081
	t-SNE	**0.949**	0.934	**0.197**	**0.250**	**0.952**	0.995	0.657	0.182	5302	**0.138**	0.658	**0.271**	**0.032**	0.049
	UMAP	0.926	0.939	0.182	0.184	0.929	0.994	**0.702**	0.172	49.0	0.266	0.646	0.346	0.070	0.045
	P-UMAP	0.643	0.770	0.022	0.012	0.645	0.960	0.352	0.046	10684	0.219	0.352	0.439	0.360	0.213
	Ours	0.928	**0.939**	0.183	0.193	0.932	**0.996**	0.687	**0.191**	39.0	0.163	**0.683**	0.303	0.066	**0.045**
Pets (3.7k)	PCA	0.785	0.885	0.096	0.023	0.790	0.928	**0.577**	0.144	**0.413**	0.187	**0.576**	0.269	0.216	0.104
	t-SNE	**0.959**	0.957	**0.684**	**0.257**	**0.976**	0.991	0.518	0.650	7700	**0.168**	0.523	0.265	**0.027**	0.031
	UMAP	0.945	**0.962**	0.667	0.196	0.969	**0.993**	0.535	**0.662**	288	0.321	0.516	0.274	0.053	**0.028**
	P-UMAP	0.723	0.835	0.085	0.016	0.728	0.900	0.489	0.137	61094	0.271	0.448	0.427	0.277	0.151
	Ours	0.946	0.956	0.649	0.206	0.966	0.989	0.465	0.593	59.6	0.189	0.468	**0.263**	0.052	0.032
Food-101 (10k)	PCA	0.727	0.890	0.058	0.011	0.729	0.934	**0.477**	0.100	**0.491**	0.173	**0.488**	0.333	0.273	0.099
	t-SNE	**0.979**	**0.958**	**0.695**	**0.254**	**0.985**	0.993	0.389	**0.652**	6565	**0.165**	0.410	0.298	**0.014**	**0.031**
	UMAP	0.964	0.955	0.671	0.158	0.975	0.993	0.355	0.612	62.1	0.209	0.369	0.332	0.035	0.034
	P-UMAP	0.551	0.670	0.024	0.003	0.552	0.802	0.092	0.044	353547	0.276	0.107	0.399	0.449	0.315
	Ours	0.956	0.955	0.646	0.170	0.965	**0.993**	0.345	0.630	43.5	0.181	0.362	**0.287**	0.042	0.034

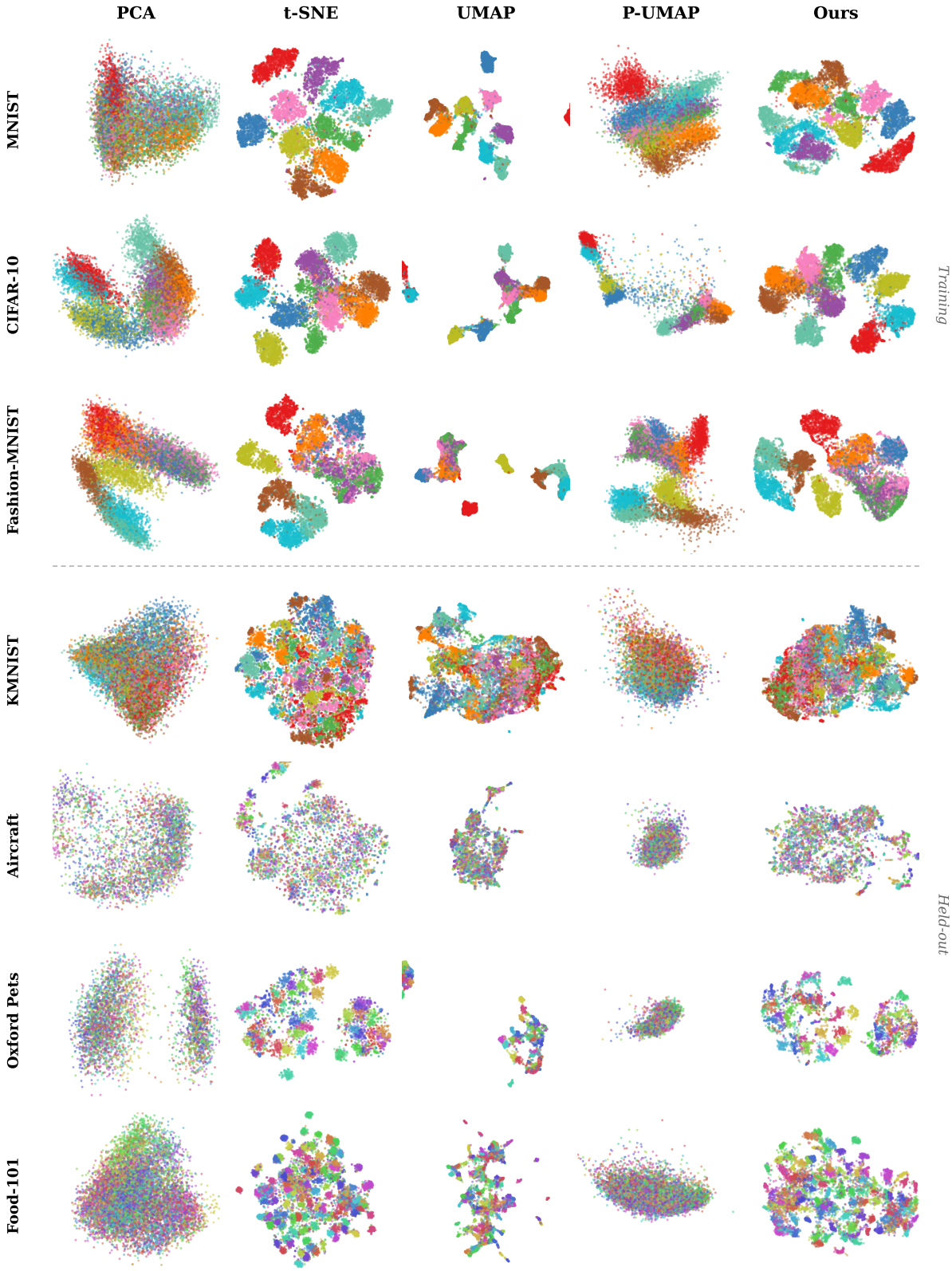

Figure 3: *2D projections for all datasets and methods. Top three rows: training datasets; bottom five rows: held-out datasets unseen during training. Points are coloured by class label.*