Parametric UMAP Embeddings for Representation and Semisupervised Learning

Tim Sainburg
timsainb@gmail.com
University of California San Diego, La Jolla, CA 92093, U.S.A.

Leland McInnes
leland.mcinnes@gmail.com
Tutte Institute for Mathematics and Computing, Ottawa, Ontario Canada

Timothy Q. Gentner
tgentner@ucsd.edu
University of California San Diego, La Jolla, CA 92093, U.S.A.

UMAP is a nonparametric graph-based dimensionality reduction algorithm using applied Riemannian geometry and algebraic topology to find low-dimensional embeddings of structured data. The UMAP algorithm consists of two steps: (1) computing a graphical representation of a data set (fuzzy simplicial complex) and (2) through stochastic gradient descent, optimizing a low-dimensional embedding of the graph. Here, we extend the second step of UMAP to a parametric optimization over neural network weights, learning a parametric relationship between data and embedding. We first demonstrate that parametric UMAP performs comparably to its nonparametric counterpart while conferring the benefit of a learned parametric mapping (e.g., fast online embeddings for new data). We then explore UMAP as a regularization, constraining the latent distribution of autoencoders, parametrically varying global structure preservation, and improving classifier accuracy for semisupervised learning by capturing structure in unlabeled data.[1]

1 Introduction

Current nonlinear dimensionality reduction algorithms can be divided broadly into nonparametric algorithms, which rely on the efficient computation of probabilistic relationships from neighborhood graphs to extract structure in large data sets (UMAP (McInnes, Healy, & Melville, 2018), t-SNE (van der Maaten & Hinton, 2008), LargeVis (Tang, Liu, Zhang, &

[1]Google Colab walkthrough.

Mei, 2016)), and parametric algorithms, which, driven by advances in deep learning, optimize an objective function related to capturing structure in a data set over neural network weights (Ding, Condon, & Shah, 2018; Ding & Regev, 2019; Hinton & Salakhutdinov, 2006; Kingma & Welling, 2013; Szubert, Cole, Monaco, & Drozdov, 2019).

In recent years, a number of parametric dimensionality reduction algorithms have been developed to wed these two classes of methods, learning a structured graphical representation of the data and using a deep neural network to capture that structure (discussed in section 3). In particular, over the past decade, several variants of the t-SNE algorithm have proposed parameterized forms of t-SNE (Bunte, Biehl, & Hammer, 2012; Gisbrecht, Lueks, Mokbel, & Hammer, 2012; Gisbrecht, Schulz, & Hammer, 2015; van der Maaten, 2009). Parametric t-SNE (van der Maaten, 2009) for example, trains a deep neural network to minimize loss over a t-SNE graph. However, the t-SNE loss function itself is not well suited to neural network training paradigms. In particular, t-SNE's optimization requires normalization over the entire data set at each step of optimization, making batch-based optimization and online learning of large data sets difficult. In contrast, UMAP is optimized using negative sampling (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Tang et al., 2016) to sparsely sample edges during optimization, making it, in principle, better suited to batch-wise training as is common in deep learning applications. Our proposed method, Parametric UMAP, brings the nonparametric graph-based dimensionality reduction algorithm UMAP into an emerging class of parametric topologically inspired embedding algorithms.

In the following section, we broadly outline the algorithm underlying UMAP to explain why our proposed algorithm, Parametric UMAP, is particularly well suited to deep learning applications. We contextualize our discussion of UMAP in t-SNE to outline the advantages that UMAP confers over t-SNE in the domain of parametric neural-network-based embedding. We then perform experiments comparing our algorithm, Parametric UMAP, to parametric and nonparametric algorithms. Finally, we show a novel extension of Parametric UMAP to semisupervised learning.

2 Parametric and Nonparametric UMAP

UMAP and t-SNE have the same goal: Given a D-dimensional data set $\mathbf{X} \in \mathbb{R}^D$, produce a d-dimensional embedding $Z \in \mathbb{R}^d$ such that points that are close together in X (e.g., x_i and x_j) are also close together in Z (z_i and z_j).

Both algorithms have the same two broad steps: first construct a graph of local relationships between data sets (see Figure 1A); then optimize an embedding in low-dimensional space that preserves the structure of the graph (see Figure 1B). The parametric approach replaces the second step of this process with an optimization of the parameters of a deep neural network

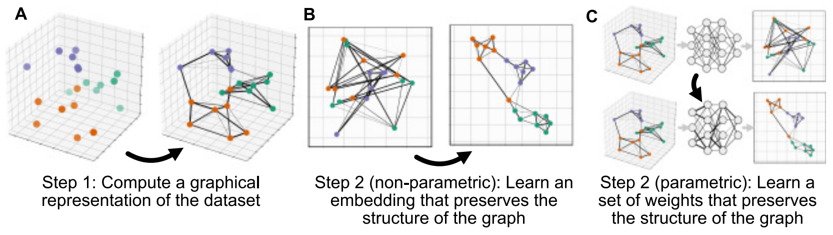
A
B
C
Step 1: Compute a graphical representation of the dataset
Step 2 (non-parametric): Learn an embedding that preserves the structure of the graph
Step 2 (parametric): Learn a set of weights that preserves the structure of the graph

Figure 1: Overview of UMAP (A $\to$ B) and Parametric UMAP (A $\to$ C).

over batches (see Figure 1C). To understand how Parametric UMAP is optimized, it is necessary to understand these steps.

2.1 Graph Construction.

2.1.1 Computing Probabilities in X. The first step in both UMAP and t-SNE is to compute a distribution of probabilities P between pairs of points in X based on the distances between points in data space. Probabilities are initially computed as local, one-directional probabilities between a point and its neighbors in data space, then symmetrized to yield a final probability representing the relationship between pairs of points.

In t-SNE, these probabilities are treated as conditional probabilities of neighborhood ($p_{i|j}^{\text{t-SNE}}$) computed using a gaussian distribution centered at x_i,

$$p_{j|i}^{\text{t-SNE}} = \frac{\exp\left(-\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\mathrm{d}(\mathbf{x}_i, \mathbf{x}_k)/2\sigma_i^2\right)}, \tag{2.1}$$

where $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)$ represents the distance between x_i and x_j (e.g., Euclidean distance) and σ_i is the standard deviation for the gaussian distribution, based on the perplexity parameter such that one standard deviation of the gaussian kernel fits a a set number of nearest neighbors in X.

In UMAP, local, one-directional probabilities ($P_{i|j}^{\text{UMAP}}$) are computed between a point and its neighbors to determine the probability with which an edge (or simplex) exists, based on an assumption that data are uniformly distributed across a manifold in a warped data space. Under this assumption, a local notion of distance is set by the distance to the kth nearest neighbor, and the local probability is scaled by that local notion of distance,

$$p_{j|i}^{\text{UMAP}} = \exp(-(\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) - \rho_i)/\sigma_i), \tag{2.2}$$

where ρ_i is a local connectivity parameter set to the distance from x_i to its nearest neighbor, and σ_i is a local connectivity parameter set to match the local distance around x_i upon its k nearest neighbors (where k is a hyperparameter).

After computing the one-directional edge probabilities for each data point, UMAP computes a global probability as the probability of either of the two local, one-directional, probabilities occurring:

$$p_{ij}^{\text{UMAP}} = \left(p_{j|i} + p_{i|j}\right) - p_{j|i}p_{i|j}. \tag{2.3}$$

In contrast, t-SNE symmetrizes the conditional probabilities as

$$p_{ij}^{\text{t-SNE}} = \frac{p_{j|i} + p_{i|j}}{2N}. \tag{2.4}$$

2.2 Graph Embedding. After constructing a distribution of probabilistically weighted edges between points in X, UMAP and t-SNE initialize an embedding in Z corresponding to each data point, where a probability distribution (Q) is computed between points as was done with the distribution (P) in the input space. The objective of UMAP and t-SNE is then to optimize that embedding to minimize the difference between P and Q.

2.2.1 Computing Probabilities in Z. In embedding space, the pairwise probabilities are computed directly without first computing local, one-directional probabilities.

In the t-SNE embedding space, the pairwise probability between two points $q_{i|j}^{\text{t-SNE}}$ is computed in a similar manner to $p_{i|j}^{\text{t-SNE}}$, but where the gaussian distribution is replaced with the fatter-tailed Student's t-distribution (with one degree of freedom), which is used to overcome the crowding problem (van der Maaten & Hinton, 2008) in translating volume differences in high-dimensional spaces to low-dimensional spaces:

$$q_{ij}^{\text{t-SNE}} = \frac{\left(1 + \|\mathbf{z}_i - \mathbf{z}_j\|^2\right)^{-1}}{\sum_{k\neq l}\left(1 + \|\mathbf{z}_k - \mathbf{z}_l\|^2\right)^{-1}}. \tag{2.5}$$

UMAP's computation of the pairwise probability q_{ij}^{UMAP} between points in the embedding space Z uses a different family of functions:

$$q_{ij}^{\text{UMAP}} = \left(1 + a\,\|z_i - z_j\|^{2b}\right)^{-1}, \tag{2.6}$$

where a and b are hyperparameters set based on a desired minimum distance between points in embedding space. Notably, the UMAP probability distribution in embedding space is not normalized, while the t-SNE distribution is normalized across the entire distribution of probabilities, meaning that the entire distribution of probabilities needs to be calculated before each optimization step of t-SNE.

2.2.2 Cost Function. Finally, the distribution of embeddings in Z is optimized to minimize the difference between Q and P.

In t-SNE, a Kullback-Leibler divergence between the two probability distributions is used, and gradient descent in t-SNE is computed over the embeddings:

$$C_{\text{t-SNE}} = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{2.7}$$

In UMAP, the cost function is cross-entropy, also optimized using gradient descent:

$$C_{\text{UMAP}} = \sum_{i \neq j} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) + (1 - p_{ij}) \log \left(\frac{1 - p_{ij}}{1 - q_{ij}} \right). \tag{2.8}$$

2.3 Attraction and Repulsion. Minimizing the cost function over every possible pair of points in the data set would be computationally expensive. UMAP and more recent variants of t-SNE both use shortcuts to bypass much of that computation. In UMAP, those shortcuts are directly advantageous to batch-wise training in a neural network.

The primary intuition behind these shortcuts is that the cost function of both t-SNE and UMAP can be broken out into a mixture of attractive forces between locally connected embeddings and repulsive forces between nonlocally connected embeddings.

2.3.1 Attractive Forces. Both UMAP and t-SNE utilize a similar strategy in minimizing the computational cost over attractive forces: they rely on an approximate nearest neighbors graph.[2] The intuition for this approach is that elements that are farther apart in data space have very small edge probabilities, which can be treated effectively as zero. Thus, edge probabilities and attractive forces only need to be computed over the nearest neighbors; non-nearest neighbors can be treated as having an edge probability of zero. Because nearest-neighbor graphs are themselves computationally expensive, approximate nearest neighbors (Dong, Moses, & Li, 2011) produce effectively similar results.

[2]UMAP requires substantially fewer nearest neighbors than t-SNE, which generally requires three times the perplexity hyperparameter (defaulted at 30 here), whereas UMAP computes only 15 neighbors by default, which is computationally less costly.

2.3.2 Repulsive Forces. Because most data points are not locally connected, we do not need to waste computation on most pairs of embeddings.

UMAP takes a shortcut motivated by the language model word2vec (Mikolov et al., 2013) and performs negative sampling over embeddings. Each training step iterates over positive, locally connected edges and randomly samples edges from the remainder of the data set, treating their edge probabilities as zero to compute cross-entropy. Because most data points are not locally connected and have a very low edge probability, these negative samples are, on average, correct, allowing UMAP to sample only sparsely over edges in the data set.

In t-SNE, repulsion is derived from the normalization of Q. A few methods for minimizing the amount of computation needed for repulsion have been developed. The first is the Barnes-Hut tree algorithm (van der Maaten, 2014), which bins the embedding space into cells and where repulsive forces can be computed over cells rather than individual data points within those cells. Similarly, the more recent interpolation-based t-SNE (FIt-SNE; Linderman, Rachh, Hoskins, Steinerberger, & Kluger, 2017, 2019) divides the embedding space into a grid and computes repulsive forces over the grid rather than the full set of embeddings.

2.4 Parametric UMAP. To summarize, both t-SNE and UMAP rely on the construction of a graph and a subsequent embedding that preserves the structure of that graph (see Figure 1). UMAP learns an embedding by minimizing cross-entropy sampled over positively weighted edges (attraction) and using negative sampling randomly over the data set (repulsion), allowing minimization to occur over sampled batches of the data set. t-SNE, meanwhile, minimizes a KL divergence loss function normalized over the entire set of embeddings in the data set using different approximation techniques to compute attractive and repulsive forces.

Because t-SNE optimization requires normalization over the distribution of embedding in projection space, gradient descent can only be performed after computing edge probabilities over the entire data set. Projecting an entire data set into a neural network between each gradient descent step would be too computationally expensive to optimize, however. The trick that Parametric t-SNE proposes for this problem is to split the data set up into large batches (e.g. 5000 data points in the original paper) that are used to compute separate graphs that are independently normalized over and used constantly throughout training, meaning that relationships between elements in different batches are not explicitly preserved. Conversely, a parametric form of UMAP, using negative sampling, can in principle be trained on batch sizes as small as a single edge, making it suitable for minibatch training needed for memory-expensive neural networks trained on the full graph over large data sets as well as online learning.

Given these design features, UMAP loss can be applied as a regularization in typical stochastic gradient descent deep learning paradigms without requiring the batching trick that Parametric T-SNE relies on. Despite this, a parametric extension to the UMAP learning algorithm has not yet been explored. Here, we explore the performance of a parametric extension to UMAP relative to current embedding algorithms and perform several experiments further extending Parametric UMAP to novel applications.[3]

3 Related Work

Beyond Parametric t-SNE and Parametric UMAP, a number of recent parametric dimensionality reduction algorithms utilizing structure-preserving constraints exist that were not compared here. This work is relevant to ours and is mentioned here to provide clarity on the current state of parametric topologically motivated and structure-preserving dimensionality reduction algorithms.

Moor et al. (2020; topological autoencoders) and Hofer, Kwitt, Niethammer, and Dixit (2019; connectivity-optimized representation learning) apply an additional topological structure-preserving loss using persistent homology over minibatches to the latent space of an autoencoder. Jia, Sun, Gao, Song, and Shi (2015; Laplacian autoencoders) similarly define an autoencoder with a local structure-preserving regularization. Mishne, Shaham, Cloninger, and Cohen (2019; Diffusion Nets) define an autoencoder extension based on diffusion maps that constrains the latent space of the autoencoder. Ding et al. (2018; scvis) and Graving and Couzin (2020; VAE-SNE) describe VAE-derived dimensionality reduction algorithms based on the ELBO objective. Duque, Morin, Wolf, and Moon (2020; geometry-regularized autoencoders) regularize an autoencoder with the PHATE (potential of heat-diffusion for affinity-based trajectory embedding) embedding algorithm (Moon et al., 2019). Szubert et al. (2019; ivis) and Robinson (2020; differential embedding networks) make use of Siamese neural network architectures with structure-preserving loss functions to learn embeddings. Pai, Talmon, Bronstein, and Kimmel (2019; DIMAL) similarly uses Siamese networks constrained to preserve geodesic distances for dimensionality reduction. Several of these parametric approaches indirectly condition neural networks (e.g., autoencoders) on nonparametric embeddings rather than directly on the loss of the algorithm, which can be applied to arbitrary embedding algorithms. We contrast indirect and direct parametric embeddings in section 5.6.

4 UMAP as a Regularization

In machine learning, regularization refers to the modification of a learning algorithm to improve generalization to new data. Here, we consider both

[3] See code implementations: Experiments: https://github.com/timsainb/ParametricUMAP_paper. Python package: https://github.com/lmcinnes/umap.

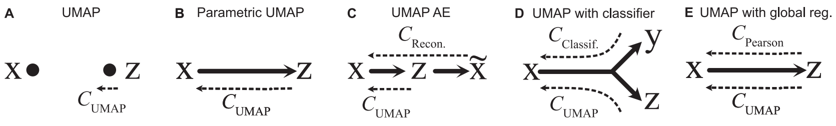

Figure 2: Variants of UMAP used in this letter. Solid lines represent neural networks. Dashed lines represent error gradients.

regularizing neural networks with UMAP loss, as well as using additional loss functions to regularize the embedding that UMAP learns. While nonparametric UMAP optimizes UMAP loss directly over embeddings (see Figure 2A), our proposed algorithm, Parametric UMAP, applies the same cost function over an encoder network (see Figure 2B). By applying additional losses, we can use regularize UMAP and UMAP to regularize additional training objectives, which we outline below.

4.1 Autoencoding with UMAP. AEs are by themselves a powerful dimensionality reduction algorithm (Hinton & Salakhutdinov, 2006). Thus, combining them with UMAP may yield additional benefits in capturing latent structure. We used an autoencoder as an additional regularization to Parametric UMAP (see Figure 2C). A UMAP/AE hybrid is simply the combination of the UMAP loss and a reconstruction loss, both applied over the network. VAEs have similarly been used in conjunction with Parametric t-SNE for capturing structure in animal behavioral data (Graving & Couzin, 2020) and combining t-SNE, which similarly emphasizes local structure, with AEs aiding in capturing more global structure over the data set (Graving & Couzin, 2020; van der Maaten & Hinton, 2008).

4.2 Semisupervised Learning. Parametric UMAP can be used to regularize supervised classifier networks, training the network on a combination of labeled data with the classifier loss and unlabeled data with UMAP loss (see Figure 2D). Semisupervised learning refers to the use of unlabeled data to jointly learn the structure of a data set while labeled data are used to optimize the supervised objective function, such as classifying images. Here, we explore how UMAP can be jointly trained as an objective function in a deep neural network alongside a classifier.

In the example in Figure 3, we show an intuitive example of semisupervised learning using UMAP over the Moons data set (Pedregosa et al., 2011). By training a Y-shaped network (see Figure 2D) on both the classifier loss over labeled data points (see Figure 3A, red and blue) and the UMAP loss over unlabeled data points (see Figure 3A, gray) jointly, the shared latent space between the UMAP and classifier network pulls apart the two moons (see Figure 3B), resulting in a decision boundary that divides cleanly between the two distributions in data space.

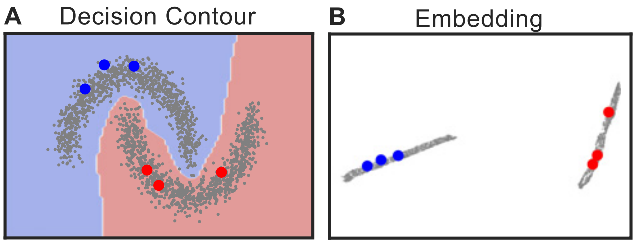

Figure 3: An example of semisupervised learning with UMAP on the moons data set.

4.3 Preserving Global Structure. An open issue in dimensionality reduction is how to balance local and global structure preservation (Becht et al., 2019; De Silva & Tenenbaum, 2003; Kobak & Linderman, 2021). Algorithms that rely on sparse nearest neighbor graphs like UMAP focus on capturing the local structure present between points and their nearest neighbors, while global algorithms, like multidimensional scaling (MDS), attempt to preserve all relationships during embedding. Local algorithms are both computationally more efficient and capture structure that is lost in global algorithms (e.g., the clusters corresponding to numbers found when projecting MNIST into UMAP). While local structure preservation captures more application-relevant structure in many data sets, the ability to additionally capture global structure is still often desirable. The approach used by nonparametric t-SNE and UMAP is to initialize embeddings with global structure-preserving embeddings such as PCA or Laplacian eigenmaps embeddings. In Parametric UMAP, we explore a different tactic, imposing global structure by jointly training on a global structure preservation loss directly.

5 Experiments

Experiments were performed comparing parametric UMAP and a UMAP/AE hybrid, to several baselines: nonparametric UMAP, nonparametric t-SNE (FIt-SNE) (Linderman et al., 2019; Poličar, Stražar, & Zupan, 2019), Parametric t-SNE, an AE, a VAE, and PCA projections. As additional baselines, we compared PHATE (nonparametric), SCVIS (parametric), and IVIS (parametric), which we described in section 3. We also compare a second nonparametric UMAP implementation that has the same underlying code as Parametric UMAP, but where optimization is performed over embeddings directly rather than neural network weights. This comparison is made to provide a bridge between the UMAP-learn implementation and Parametric UMAP, to control for any potential implementation differences. Parametric t-SNE, Parametric UMAP, the AE, VAE, and the UMAP/AE hybrid use the same neural network architectures and optimizers within each data set (described in supplemental materials).

We used the common machine learning benchmark data sets MNIST, FMNIST, and CIFAR-10 alongside two real-world data sets in areas where UMAP has proven a useful tool for dimensionality reduction: a single-cell retinal transcriptome data set (Macosko et al., 2015) and a bioacoustic data set of Cassin's vireo song, recorded in the Sierra Nevada mountains (Hedley, 2016a, 2016b).

5.1 Embeddings. We first confirm that Parametric UMAP produces embeddings that are of a similar quality to nonparametric UMAP. To quantitatively measure the quality of embeddings, we compared embedding algorithms on several metrics across data sets. We compared each method/data set on 2D and 64D projections.[4] Each metric is explained in detail in the supplemental materials. The 2D projection of each data set/method is shown in Figure 4. The results are given in Supplementary Figures 1–6 and Supplementary Tables 2–7 and summarized below.

5.1.1 Trustworthiness. Trustworthiness (see supplementary equation 1 and Venna & Kaski, 2006) is a measure of how much of the local structure of a data set is preserved in a set of embeddings. In 2D, we observe that each of the UMAP algorithms performs similarly in trustworthiness, with t-SNE being slightly more trustworthy in each data set (see Figure 5, Supplementary Figure 1, and Supplementary Table 2. At 64D, PCA, AE, VAE, and Parametric t-SNE are most trustworthy in comparison to each UMAP implementation, possibly reflecting the more approximate repulsion (negative sampling) used by UMAP.

5.1.2 Area under the Curve (AUC) of R_{NX}. To compare embeddings across scales (both local and global neighborhoods), we computed the AUC of R_{NX} for each embedding (Lee, Peluffo-Ordóñez, & Verleysen, 2015), which captures the agreement across K-ary neighborhoods, weighting nearest neighbors as more important than farther neighbors. In 2D we find that Parametric and nonparametric UMAP perform similarly, while t-SNE has the highest AUC. At 64D, Parametric and nonparametric UMAP again perform similarly, with PCA having the higher AUC.

5.1.3 KNN-Classifier. A KNN-classifier is used as a baseline to measure supervised classification accuracy based on local relationships in

[4]Where possible. In contrast with UMAP, Parametric UMAP, and Parametric t-SNE, Barnes Huts t-SNE can only embed in two or three dimensions (van der Maaten, 2014), and while FIt-SNE can in principle scale to higher dimensions (Linderman et al., 2019), embedding in more than two dimensions is unsupported in both the official implementation (KlugerLab, 2020) and openTSNE (Poličar et al., 2019)

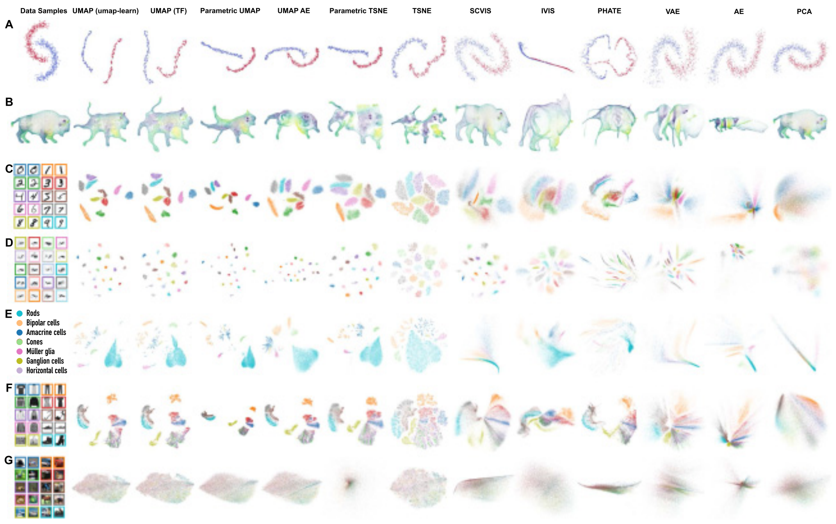

Figure 4: Comparison of projections from multiple data sets using UMAP, UMAP in tensorflow, parametric UMAP, parametric UMAP with an autoencoder loss, parametric t-SNE, t-SNE, SCVIS, IVIS, PHATE, a VAE, an AE, and PCA. (a) Moons. (B) 3D buffalo. (c) MNIST, (d) Cassin's vireo song segments, (e) Mouse retina single-cell transcriptomes. (f) Fashion MNIST, (g) CIFAR10. The Cassin's vireo data set uses a dynamic time warping loss and an LSTM network for the encoder and decoder for the neural networks. The image data sets use a convnet for the encoder and decoder for the neural networks. The bison examples use a t-SNE perplexity of 500 and 150 nearest neighbors in UMAP to capture more global structure.

embeddings. We find KNN-classifier performance largely reflects trustworthiness (see Figure 5, Supplementary Figures 3 and 4, and Supplementary Tables 4 and 5). In 2D, we observe a broadly similar performance between UMAP and t-SNE variants, each of which is substantially better than the PCA, AE, or VAE projections. At 64 dimensions UMAP projections are similar but in some data sets (FMNIST, CIFAR-10) slightly under-perform PCA, AE, VAE, and Parametric t-SNE.

5.1.4 Silhouette Score. Silhouette score measures how clustered a set of embeddings is given ground truth labels. In 2D, across data sets, we tend to see a better silhouette score for UMAP and Parametric UMAP projections than t-SNE and Parametric t-SNE, which are more clustered than PCA in all cases but CIFAR-10, which shows little difference from PCA (see Figure 5, Supplementary Figure 5 and Supplementary Table 5). The clustering of each data set can also be observed in Figure 4, where t-SNE and

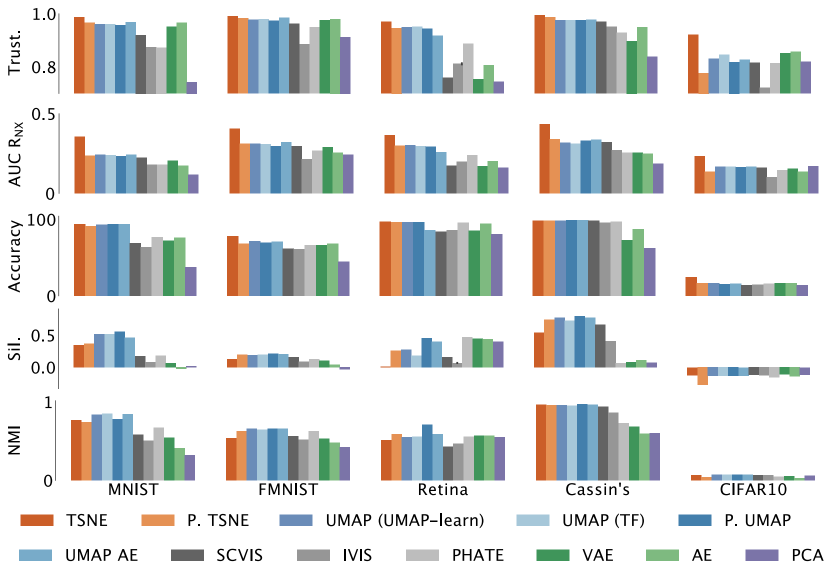

Figure 5: Embedding metrics for 2D projections. Full results are given in the appendix. Accuracy is shown for KNN (k = 1).

Parametric t-SNE are more spread out within the cluster than UMAP. In 64D projections, we find the silhouette score of Parametric t-SNE is near or below that of PCA, which is lower than UMAP-based methods. We note, however, that the poor performance of Parametric t-SNE may reflect setting the degrees-of-freedom (α) at $d - 1$, which is only one of three parameterization schemes that van der Maaten (2009) suggests. A learned degrees-of-freedom parameter might improve performance for parametric t-SNE at higher dimensions.

5.1.5 Clustering. To compare clustering directly across embeddings, we performed k-means clustering over each latent projection and compared each embedding's clustering on the basis of the normalized mutual information (NMI) between clustering schemes (see Figure 5, Supplemental Figure 6, and Supplementary Table 6). In both the 2D and 64D projections, we find that NMI corresponds closely to the silhouette score. UMAP and t-SNE show comparable clustering in 2D, both well above PCA in most data sets. At 64D, each UMAP approach shows superior performance over t-SNE.

5.2 Training and Embedding Speed.

5.2.1 Training Speed. Optimization in nonparametric UMAP is not influenced by the dimensionality of the original data set; the dimensionality

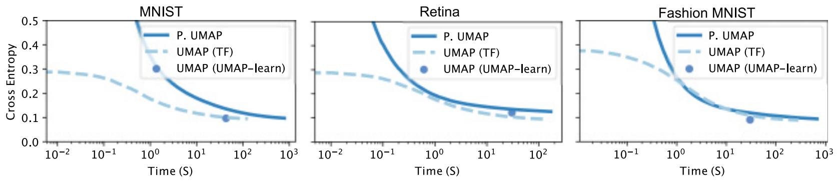

Figure 6: Training times comparison between UMAP and Parametric UMAP. All results were obtained with up to 32 threads on a machine with 2 AMD EPYC Rome 7252 8-core CPU running at 3.1 GHz and a Quadro RTX 6000.

comes into play only in computing the nearest-neighbors graph. In contrast, training speeds for Parametric UMAP are variable based on the dimensionality of data and the architecture of the neural network used. The dimensionality of the embedding does not have a substantial effect on speed. In Figure 6, we show the cross-entropy loss over time for Parametric and nonparametric UMAP, for the MNIST, Fashion MNIST, and Retina data sets. Across each data set, we find that nonparametric UMAP reaches a lower loss more quickly than Parametric UMAP but that Parametric UMAP reaches a similar cross-entropy within an order of magnitude of time. Thus, Parametric UMAP can train more slowly than nonparametric UMAP, but training times remain within a similar range, making Parametric UMAP a reasonable alternative to nonparametric UMAP in terms of training time.

5.2.2 Embedding and Reconstruction Speed. A parametric mapping allows embeddings to be inferred directly from data, resulting in a quicker embedding than nonparametric methods. The speed of embedding is especially important in signal processing paradigms where near-real-time embedding speeds are necessary. For example in brain-machine interfacing, bioacoustics, and computational ethology, fast embedding methods like PCA or deep neural networks are necessary for real-time analyses and manipulations; thus, deep neural networks are increasingly being used (Brown & De Bivort, 2018; Pandarinath et al., 2018; Sainburg, Thielk, & Gentner, 2019). Here, we compare the embedding speed of a held-out test sample for each data set, as well as the speed of reconstruction of the same held-out test samples.

Broadly, we observe similar embedding times for the nonparametric t-SNE and UMAP methods, which are several orders of magnitude slower than the parametric methods, where embeddings are direct projections into the learned networks (see Figure 7). Because the same neural networks are used across the different parametric UMAP and t-SNE methods, we show only Parametric UMAP in Figure 7, which is only slightly slower than PCA, making it a viable candidate for fast embedding where PCA is currently used. Similarly, we compared parametric and nonparametric UMAP reconstruction speeds (see Supplemental Figure 7). With the network

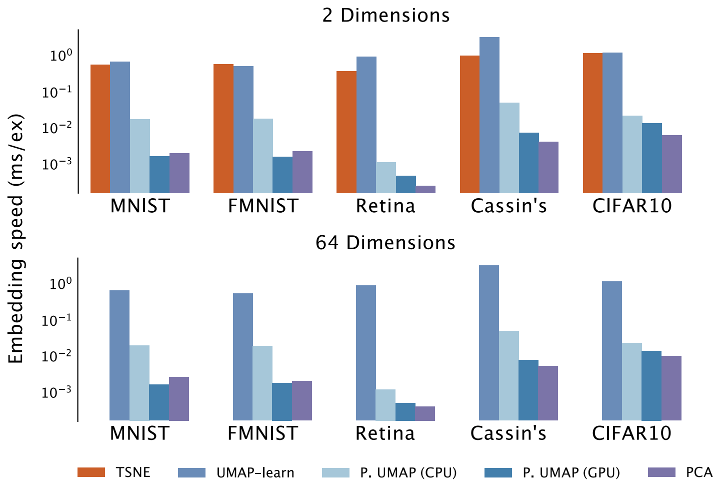

Figure 7: Comparison of embedding speeds using Parametric UMAP and other embedding algorithms on a held-out testing data set. Embeddings were performed on the same machine as Figure 6. Values shown are the median times over 10 runs.

architectures we used, reconstructions of Parametric UMAP are orders of magnitude faster than nonparametric UMAP, and slightly slower, but within the same order of magnitude as PCA.

5.3 Capturing Additional Global Structure in Data. To capture additional global structure we added a naive global structure preservation loss to Parametric UMAP, maximizing the Pearson correlation within batches between pairwise distances in embedding and data spaces,

$$C_{\text{Pearson}} = -\frac{\text{cov}(d_X, d_Z)}{\sigma_{d_X}\sigma_{d_Z}}, \tag{5.1}$$

where $\text{cov}(X, Y)$ is the covariance of data and embeddings and σ_X and σ_Z are the standard deviations of the data and embeddings. The same notion of pairwise distance correlation has previously been used directly as a metric for global structure preservation (Becht et al., 2019; Kobak & Linderman, 2021).

The weight of this additional loss can be used to dictate the balance between capturing global and local structure in the data set. In Figure 8, we apply this loss at four different weights, ranging from only UMAP (left) to primarily global correlation (right). As expected, we observe that as we weight C_{Pearson} more heavily, the global correlation (measured as the correlation of the distance between pairs of points in embedding and data space)

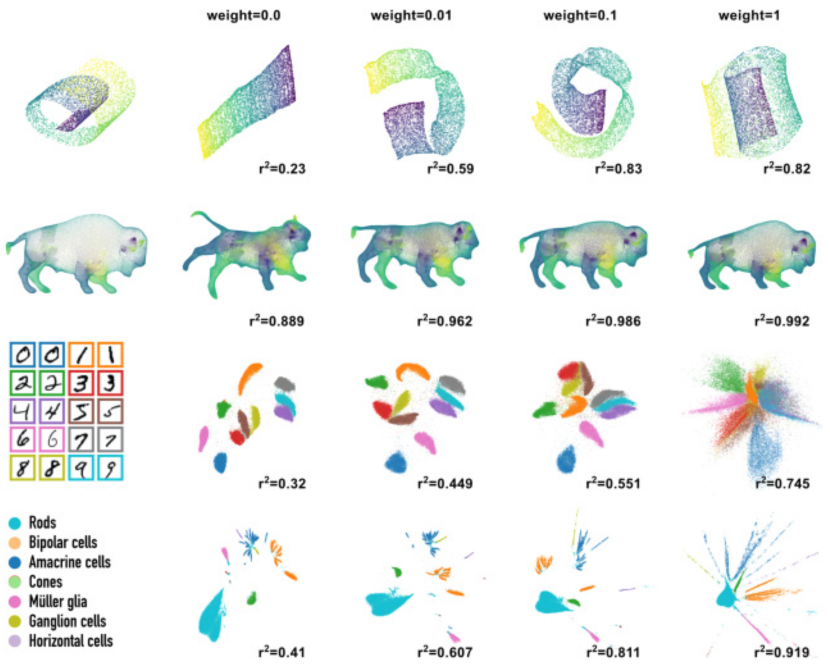

Figure 8: Global loss applied to Parametric UMAP embeddings with different weights. r^2 is the correlation between pairwise distances in data space and embedding space.

increases (indicated in each panel). Notably, when a small weight is used with each data set, local structure is largely preserved while substantially improving global correlation.

In Figure 9, we show the global distance correlation plotted against two local structure metrics (silhouette score and trustworthiness) for the MNIST and Macosko et al. (2015) data sets corresponding to the projections shown in Figure 8 in relation to each embedding from Figure 4. In addition, we compared TriMap (Amid & Warmuth, 2019), a triplet-loss-based embedding algorithm designed to capture additional global structure, by preserving the relative distances of triplets of data samples. We also compared minimum distortion embedding (MDE), which comprises two separate embedding functions: a local embedding algorithm that preserves relationships between neighbors similar to UMAP and t-SNE and a global embedding algorithm that preserves pairwise distances similar to MDS.

Broadly, with Parametric UMAP, we can observe the trade-off between captured global and local structure with the weight of C_{Pearson} (light blue line in each panel of Figure 9). We observe that adding this loss can increase the amount of global structure captured while preserving much of the local

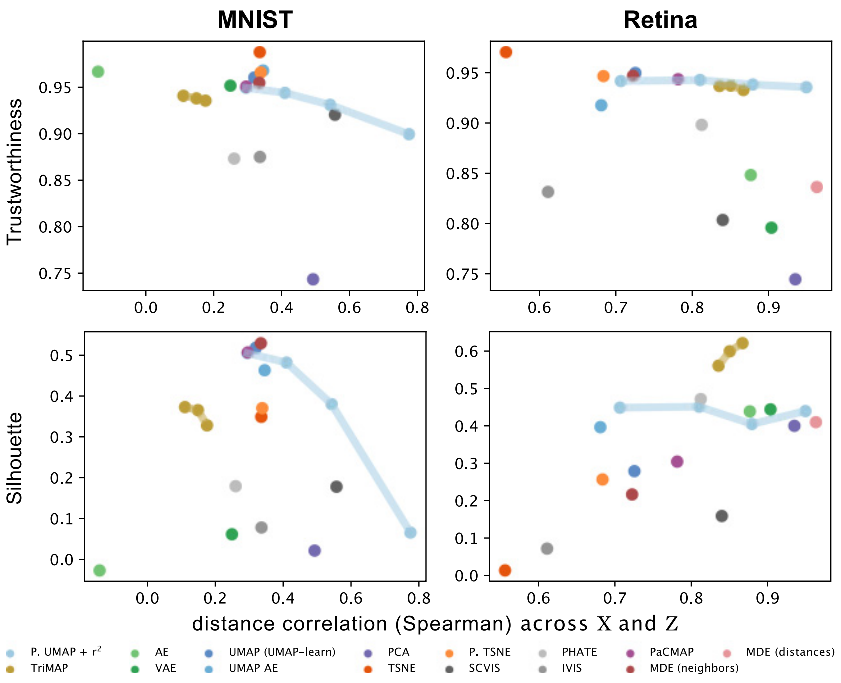

Figure 9: Comparison of pairwise global and local relationship preservation across embeddings for MNIST and Macosko et al. (2015). Local structure metrics are silhouette score and trustworthiness. Global structure metric is Spearman correlation of distances in X and Z. Connected lines are for the different weights of the correlation loss from Figure 8 in Parametric UMAP and the λ parameter in TriMap (50, 500, and 5000). MDE (distances) is not given for MNIST because of memory issues (on 512 GB RAM).

structure, as indicated by the distance to the top right corner of each panel in Figure 9, which reflects the simultaneous capture of global and local relationships, relative to each other embedding algorithm.

5.4 Autoencoding with UMAP. The ability to reconstruct data from embeddings can aid in understanding the structure of nonlinear embeddings and allow for manipulation and synthesis of data based on the learned features of the data set. We compared the reconstruction accuracy across each method, which had inverse-transform capabilities ($Z \to X$), as well as the reconstruction speed across the neural network–based implementations to nonparametric implementations and PCA. In addition, we performed latent algebra on Parametric UMAP embeddings both with and without an

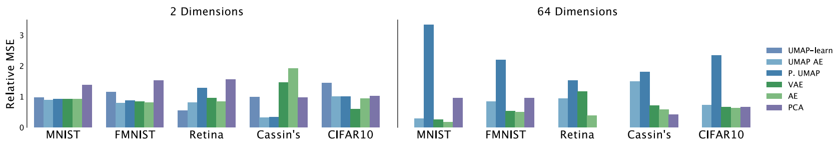

Figure 10: Reconstruction accuracy measured as mean squared error (MSE). MSE is shown relative to each data set (setting mean at 1).

autoencoder regularization and found that reconstructed data can be linearly manipulated in complex feature space.

5.4.1 Reconstruction Accuracy. We measured reconstruction accuracy as mean squared error (MSE) across each data set (see Figure 10 and Supplementary Table 7). In two dimensions, we find that Parametric UMAP typically reconstructs better than nonparametric UMAP, which in turn performs better than PCA. In addition, the autoencoder regularization slightly improves reconstruction performance. At 64 dimensions, the AE regularized Parametric UMAP is generally comparable to the AE and VAE and performs better than Parametric UMAP without autoencoder regularization. The nonparametric UMAP reconstruction algorithm is not compared at 64 dimensions because it relies on an estimation of Delaunay triangulation, which does not scale well with higher dimensions.

5.4.2 Latent Features. Previous work shows that parametric embedding algorithms such as AEs (e.g., variational autoencoders) linearize complex data features in latent space—for example, the presence of a pair of sunglasses in pictures of faces (Radford, Metz, & Chintala, 2015; Sainburg et al., 2018; White, 2016). Here, we performed latent-space algebra and reconstructed manipulations on Parametric UMAP latent space to explore whether UMAP does the same.

To do so, we use the CelebAMask-HQ data set which contains annotations for 40 different facial features over a highly structured data set of human faces. We projected the data set of faces into a CNN autoencoder architecture based on the architecture defined in Huang, Liu, Belongie, and Kautz (2018). We trained the network first using UMAP loss alone (Parametric UMAP) and then using the joint UMAP and AE loss (see Figure 11). We then fit an OLS regression to predict the latent projections of the entire data set using the 40 annotated features (e.g., hair color, presence of beard, smiling). The vectors corresponding to each feature learned by the linear model were then treated as feature vectors in latent space and added and subtracted from projected images, then passed through the decoder to observe the resulting image (as in Sainburg et al., 2018).

We find that complex latent features are linearized in latent space when the network is trained with UMAP loss alone as well as when the network

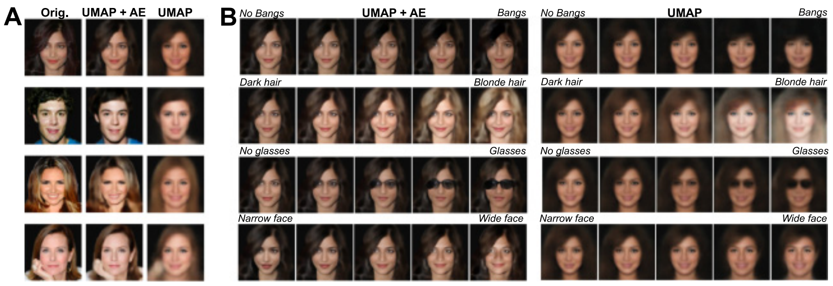

Figure 11: Reconstruction and interpolation. (A) Parametric UMAP reconstructions of faces from a holdout testing data set. (B) The same networks, adding latent vectors corresponding to image features.

is trained with AE loss. For example, in the third set of images in Figure 10, a pair of glasses can be added or removed from the projected image by adding or subtracting its corresponding latent vector.

5.5 Semisupervised Learning. Real-word data sets often comprise a small number of labeled data and a large number of unlabeled data. Semisupervised learning (SSL) aims to use the unlabeled data to learn the structure of the data set, aiding a supervised learning algorithm in making decisions about the data. Current SOTA approaches in many areas of supervised learning such as computer vision rely on deep neural networks. Likewise, semisupervised learning approaches modify supervised networks with structure-learning loss using unlabeled data. Parametric UMAP, being a neural network that learns structure from unlabeled data, is well suited to semisupervised applications. Here, we determine the efficacy of UMAP for semisupervised learning by comparing a neural network jointly trained on classification and UMAP (see Figure 2D) with a network trained on classification alone using data sets with varying numbers of labeled data.

We compared data sets ranging from highly structured (MNIST) to unstructured (CIFAR-10) in UMAP using a naive distance metric in data space (e.g., Euclidean distance over images). For image data sets, we used a deep convolutional neural network (CNN), which performs with relatively high accuracy for CNN classification on the fully supervised networks (see Supplementary Table 8 based on the CNN13 architecture commonly used in SSL (Oliver, Odena, Raffel, Cubuk, & Goodfellow, 2018). For the birdsong data set, we used a BLSTM network, and for the retina data set, we used a densely connected network.

5.5.1 Naive UMAP Embedding. For data sets where structure is learned in UMAP (e.g., MNIST, FMNIST), we expect that regularizing a classifier network with UMAP loss will aid the network in labeling data by learning

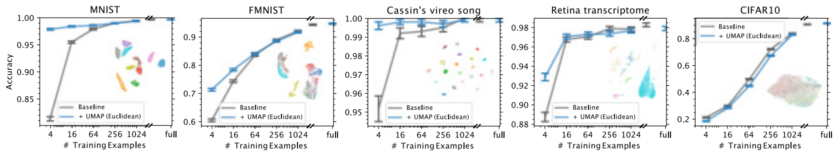

Figure 12: Baseline classifier with an additional UMAP loss with different numbers of labeled training examples. Nonparametric UMAP projections of the UMAP graph being jointly trained are shown at the bottom right of each panel. Error bars show SEM.

the structure of the data set from unlabeled data. To test this, we compared a baseline classifier to a network jointly trained on classifier loss and UMAP loss. We first trained the baseline classifier to asymptotic performance on the validation data set, then, using the pretrained weights from the baseline classifier, trained a Y-shaped network (see Figure 2D) jointly on UMAP over Euclidean distances and a classifier loss over the data set. We find that for each data set where categorically relevant structure is found in latent projections of the data sets (MNIST, FMNIST, birdsong, retina), classifications are improved in the semisupervised network over the supervised network alone, especially with smaller numbers of training examples (see Figure 12 and Supplementary Table 8). In contrast, for CIFAR-10, the additional UMAP loss impairs performance in the classifier.

5.5.2 Consistency Regularization and Learned Invariance Using Data Augmentation. Several current SOTA SSL approaches employ a technique called consistency regularization (Sajjadi, Javanmardi, & Tasdizen, 2016) that is, training a classifier to produce the same predictions with unlabeled data that have been augmented and data that have not been augmented (Berthelot et al., 2020; Sohn et al., 2020). In a similar vein, for each image data set, we train the network to preserve the structure of the UMAP graph when data have been augmented. We computed a UMAP graph over unaugmented data and, using augmented data, trained the network jointly using classifier and UMAP loss, teaching the network to learn to optimize the same UMAP graph, invariant to augmentations in the data. We observe a further improvement in network accuracy for MNIST and FMNIST over the baseline and the augmented baseline (see Figure 13, left and Supplementary Table 8). For the CIFAR-10 data set, the addition of the UMAP loss, even over augmented data, reduces classification accuracy.

5.5.3 Learning a Categorically Relevant UMAP Metric Using a Supervised Network. It is unsurprising that UMAP confers no improvement for the CIFAR-10 data set, as UMAP computed over the pixel-wise Euclidean distance between images in the CIFAR-10 data set does not capture very much

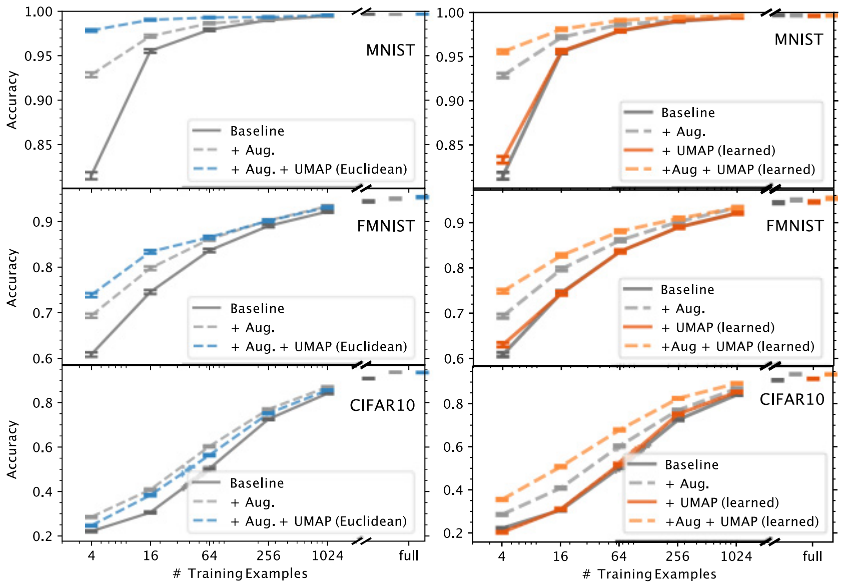

Figure 13: Comparison of baseline classifier, augmentation, and augmentation with an additional UMAP loss (left). SSL using UMAP over the learned latent graph, computed over latent activations in the classifier (right).

categorically relevant structure in the data set. Because no common distance metric over CIFAR-10 images is likely to capture such structure, we consider using supervision to learn a categorically relevant distance metric for UMAP. We do so by training on a UMAP graph computed using distance over latent activations in the classifier network (as in Carter, Armstrong, Schubert, Johnson, & Olah, 2019), where categorical structure can be seen in UMAP projection (see Figure 14). The intuition is that training the network with unlabeled data to capture distributional structure within the network's learned categorically relevant space will aid in labeling new data.

We find that in all three data sets, without augmentation, the addition of the learned UMAP loss confers little to no improvement in classification accuracy over the data (see Figure 13, right, and Supplementary Table 8). When we look at nonparametric projections of the graph over latent activations, we see that the learned graph largely conforms to the network's categorical decision making (e.g., Figure 14 predictions versus ground truth). In contrast, with augmentation, the addition of the UMAP loss improves performance in each data set, including CIFAR-10. This contrast in improvement demonstrates that training the network to learn a distribution in a categorically relevant space that is already intrinsic to the network does not confer any additional information that the network can use in classification. Training the network to be invariant toward augmentations

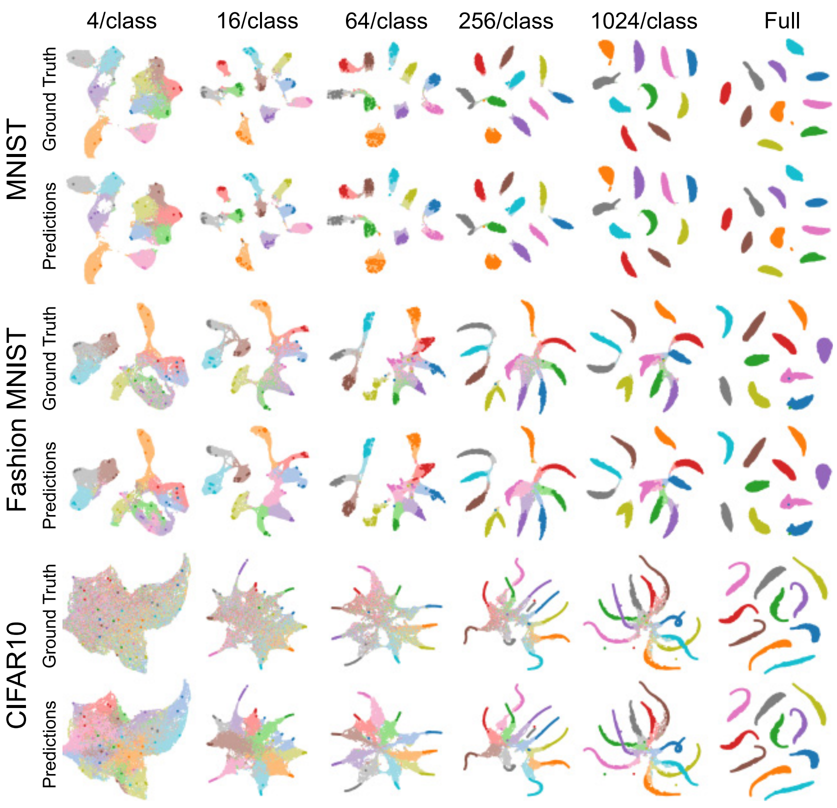

Figure 14: Nonparametric UMAP projections of activations in the last layer of a trained classifier for MNIST, FMNIST, and CIFAR-10. For each data set, the top row shows the ground truth labels and the model's predictions in a light color map. On top of each projection, the labeled data points used for training are shown in a darker color map.

in the data, however, does aid in regularizing the classifier, more in line with directly training the network on consistency in classifications (Sajjadi et al., 2016).

5.6 Comparisons with Indirect Parametric Embeddings. In principle, any embedding technique can be implemented parametrically by training a parametric model (e.g., a neural network) to predict embeddings from the original high-dimensional data (as in Duque et al., 2020). However, such a parametric embedding is limited in comparison to directly optimizing the algorithm's loss function. Parametric UMAP optimizes directly over the structure of the graph with respect to the architecture of the network as well as additional constraints (e.g., additional losses). In contrast, training a

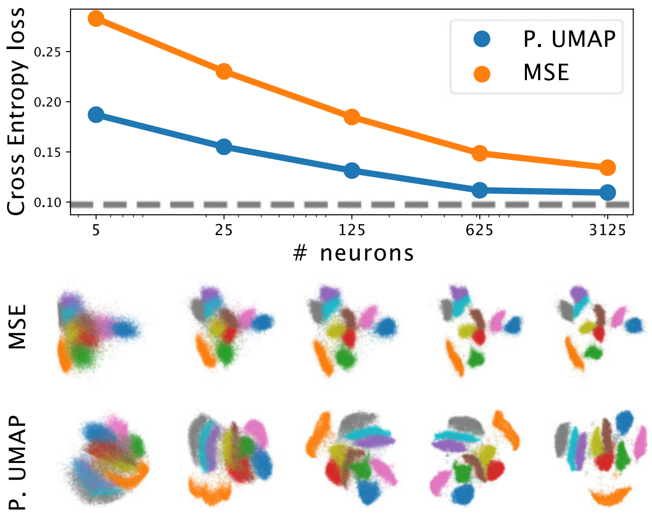

Figure 15: Top: Cross-entropy loss for one-hidden-layer instance of Parametric UMAP versus a neural network trained to predict nonparametric embeddings using MSE on MNIST. The same network architectures are used in each case. The x-axis varies the number of neurons in the network's single hidden layer layer. The dashed gray line is the loss for the nonparametric embedding. Bottom: Projections corresponding to the losses shown in the panel above.

neural network to predict nonparametric embeddings does not take additional constraints into account.

To exemplify this, in Figure 15 we compare Parametric UMAP to a neural network trained to predict nonparametric embeddings by minimizing MSE when the number of neurons in the network is limited. In the case of parametric UMAP, the objective of the network is to come up with the best embedding of the UMAP graph that it can, given the constraints of the architecture of the network. In the indirect/MSE case, information about the structure of the graph is available only through an intermediary, the nonparametric embedding; thus, this method cannot be optimized to learn an embedding of the data that best preserve the structure of the graph. In other words, the indirect method is not optimizing the embedding of the graph with respect to additional constraints. Instead, it is minimizing the distance between two sets of embeddings. The weighted graph is an intermediate topological representation (notably of no specific dimensionality) and is the best representation of the data under UMAP's assumptions. The process of embedding the data in a fixed dimensional space is necessarily a lossy one. Optimizing over the graph directly avoids this loss. This issue also applies when incorporating additional losses (e.g., a classifier loss or autoencoder loss) to indirect embeddings.

6 Discussion

In this letter, we propose a novel parametric extension to UMAP. This parametric form of UMAP produces embeddings similar to those of nonparametric UMAP, with the added benefit of a learned mapping between data space and embedding space. We demonstrated the utility of this learned mapping on several downstream tasks. We showed that parametric relationships can be used to improve inference times for embeddings and reconstructions by orders of magnitude while maintaining similar embedding quality to nonparametric UMAP. Combined with a global structure preservation loss, Parametric UMAP captures additional global relationships in data, outperforming methods where global structure is only imposed upon initialization (e.g., initializing with PCA embeddings). Combined with an autoencoder, UMAP improves reconstruction quality and allows for the reconstruction of high-dimensional UMAP projections. We also show that Parametric UMAP projections linearize complex features in latent space. Parametric UMAP can be used for semisupervised learning, improving training accuracy on data sets where small numbers of training exemplars are available. We showed that UMAP loss applied to a classifier improves semisupervised learning in real-world cases where UMAP projections carry categorically relevant information (such as stereotyped birdsongs or single-cell transcriptomes), but not in cases where categorically relevant structure is not present (such as CIFAR-10). We devised two downstream approaches based around learned categorically relevant distances and consistency regularization that show improvements on these more complex data sets. Parametric embedding also makes UMAP feasible in fields where dimensionality reduction of continuously generated signals plays an important role in real-time analysis and experimental control.

A number of future directions and extensions to our approach have the potential to further improve upon our results in dimensionality reduction and its various applications. For example, to improve global structure preservation, we jointly optimized over the Pearson correlation between data and embeddings. Using notions of global structure beyond pairwise distances in data space (such as global UMAP relationships or higher-dimensional simplices) may capture additional structure in data. Similarly, one approach we used to improve classifier accuracy relied on obtaining a categorically relevant metric, defined as the Euclidean distance between activation states of the final layer of a classifier. Recent work (e.g., as discussed and proposed in Schulz, Hinder, & Hammer, 2019) have explored methods for more directly capturing class information in the computation of distance, such as using the Fisher metric to capture category- and decision-relevant structure in classifier networks. Similar metrics may prove to further improve semisupervised classifications with Parametric UMAP.

Acknowledgments

This work was supported by NIH 5T32MH020002-20 to T.S. and 5R01DC018055-02 to T.G. We also thank Kyle McDonald for making available his translation of Parametric t-SNE to Tensorflow/Keras, which we used as a basis for our own implementation.

References

Amid, E., & Warmuth, M. K. (2019). *Trimap: Large-scale dimensionality reduction using triplets.* arXiv:1910.00204.

Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W., Ng, L. G., . . . Newell, E. W. (2019). Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology*, *37*(1), 38–44. https://doi.org/10.1038/nbt.4314

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., & Raffel, C. (2020). Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *Proceedings of the International Conference on Learning Representations.* https://openreview.net/forum?id=HklkeR4KPB

Brown, A. E., & De Bivort, B. (2018). Ethology as a physical science. *Nature Physics*, *14*(7), 653–657. https://doi.org/10.1038/s41567-018-0093-0

Bunte, K., Biehl, M., & Hammer, B. (2012). A general framework for dimensionality-reducing data visualization mapping. *Neural Computation*, *24*(3), 771–804. https://doi.org/10.1162/NECO_a_00250

Carter, S., Armstrong, Z., Schubert, L., Johnson, I., & Olah, C. (2019). Activation atlas. *Distill*, *4*(3), e15.

De Silva, V., & Tenenbaum, J. B. (2003). Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, & K. Overmayer (Eds.), *Advances in neural information processing systems*, *15* (pp. 721–728). Red Hook, NY: Curran.

Ding, J., Condon, A., & Shah, S. P. (2018). Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature Communications*, *9*(1), 1–13. https://doi.org/10.1038/s41467-018-04368-5, PubMed: 29784946

Ding, J., & Regev, A. (2019). *Deep generative model embedding of single-cell RNA-Seq profiles on hyperspheres and hyperbolic spaces*. bioRxiv:853457.

Dong, W., Moses, C., & Li, K. (2011). Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web* (pp. 577–586). New York: ACM. https://doi.org/10.1145/1963405.1963487

Duque, A. F., Morin, S., Wolf, G., & Moon, K. R. (2020). *Extendable and invertible manifold learning with geometry regularized autoencoders.* arXiv:2007.07142.

Gisbrecht, A., Lueks, W., Mokbel, B., & Hammer, B. (2012). Out-of-sample kernel extensions for nonparametric dimensionality reduction. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence, and Machine Learning.* https://doi.org/10.1109/TNNLS.2015.2512277

Gisbrecht, A., Schulz, A., & Hammer, B. (2015). Parametric nonlinear dimensionality reduction using kernel t-SNE. *Neurocomputing*, *147*, 71–82. https://doi.org/10.1016/j.neucom.2013.11.045

Graving, J. M., & Couzin, I. D. (2020). *Vae-SNE: A deep generative model for simultaneous dimensionality reduction and clustering*. bioRxiv. https://doi.org/10.1101/2020.07.17.207993

Hedley, R. W. (2016a). Complexity, predictability and time homogeneity of syntax in the songs of Cassin's Vireo (*Vireo cassinii*). *PLOS One*, *11*(4), e0150822. https://doi.org/10.1371/journal.pone.0150822, PubMed: 27050537

Hedley, R. W. (2016b). Composition and sequential organization of song repertoires in Cassin's Vireo (*Vireo cassinii*). *Journal of Ornithology*, *157*(1), 13–22. https://doi.org/10.1007/s10336-015-1238-x

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647, PubMed: 16873662

Hofer, C., Kwitt, R., Niethammer, M., & Dixit, M. (2019). Connectivity-optimized representation learning via persistent homology. In *Proceedings of the International Conference on Machine Learning* (pp. 2751–2760). Berlin: Springer.

Huang, X., Liu, M.-Y., Belongie, S., & Kautz, J. (2018). Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision* (pp. 172–189). Berlin: Springer. https://doi.org/10.1007/978-3-030-01219-9_11

Jia, K., Sun, L., Gao, S., Song, Z., & Shi, B. E. (2015). Laplacian auto-encoders: An explicit learning of nonlinear data manifold. *Neurocomputing*, *160*, 250–260. https://doi.org/10.1016/j.neucom.2015.02.023

Kingma, D. P., & Welling, M. (2013). *Auto-encoding variational Bayes*. arXiv:1312.6114.

KlugerLab. (2020). *Open source survey.* https://github.com/KlugerLab/FIt-SNE, commit=4f57d6a0e4c030202a07a60bc1bb1ed1544bf679.

Kobak, D., & Linderman, G. C. (2021). Initialization is critical for preserving global data structure in both t-SNE and UMAP. *Nature Biotechnology*, 1–2. https://doi.org/10.1038/s41587-020-00809-z

Lee, J. A., Peluffo-Ordóñez, D. H., & Verleysen, M. (2015). Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. *Neurocomputing*, *169*, 246–261. https://doi.org/10.1016/j.neucom.2014.12.095

Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., & Kluger, Y. (2017). *Efficient algorithms for t-distributed stochastic neighborhood embedding.* arXiv:1712.09005. https://doi.org/10.1038/s41592-018-0308-4

Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., & Kluger, Y. (2019). Fast interpolation-based t-SNE for improved visualization of single-cell RNA-Seq data. *Nature Methods*, *16*(3), 243–245. https://doi.org/10.1038/s41592-018-0308-4

Macosko, E. Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., . . . McCarroll, S. A. (2015). Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, *161*(5), 1202–1214. https://doi.org/10.1016/j.cell.2015.05.002, PubMed: 26000488

McInnes, L., Healy, J., & Melville, J. (2018). *Umap: Uniform manifold approximation and projection for dimension reduction*. arXiv:1802.03426.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *26* (pp. 3111–3119). Red Hook, NY: Curran.

Mishne, G., Shaham, U., Cloninger, A., & Cohen, I. (2019). Diffusion nets. *Applied and Computational Harmonic Analysis*, *47*(2), 259–285.

Moon, K. R., van Dijk, D., Wang, Z., Gigante, S., Burkhardt, D. B., Chen, W. S., . . . Krishnaswamy (2019). Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, *37*(12), 1482–1492. https://doi.org/10.1038/s41587-019-0336-3

Moor, M., Horn, M., Rieck, B., & Borgwardt, K. (2020). Topological autoencoders. In *Proceedings of the International Conference on Machine Learning*.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., & Goodfellow, I. (2018). Realistic evaluation of deep semi-supervised learning algorithms. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*, *31* (pp. 3235–3246). Red Hook, NY: Curran.

Pai, G., Talmon, R., Bronstein, A., & Kimmel, R. (2019). DIMAL: Deep isometric manifold learning using sparse geodesic sampling. In *Proceedings of the 2019 IEEE Winter Conference on Applications of Computer Vision* (pp. 819–828). Piscataway, NJ: IEEE.

Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., . . . Sussilo, D. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, *15*(10), 805–815. https://doi.org/10.1038/s41592-018-0109-9

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Cournapeau, D. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Poličar, P. G., Stražar, M., & Zupan, B. (2019). *openTSNE: A modular Python library for t-SNE dimensionality reduction and embedding*. bioRxiv. https://www.biorxiv.org/content/early/2019/08/13/731877

Radford, A., Metz, L., & Chintala, S. (2015). *Unsupervised representation learning with deep convolutional generative adversarial networks.* arXiv:1511.06434.

Robinson, I. (2020). *Interpretable visualizations with differentiating embedding networks.* arXiv:2006.06640.

Sainburg, T., Thielk, M., & Gentner, T. Q. (2019). *Latent space visualization, characterization, and generation of diverse vocal communication signals.* doi:https://doi.org/10.1101/870311

Sainburg, T., Thielk, M., Theilman, B., Migliori, B., & Gentner, T. (2018). *Generative adversarial interpolative autoencoding: Adversarial training on latent space interpolations encourage convex latent distributions.* arXiv:1807.06650.

Sajjadi, M., Javanmardi, M., & Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*, *29* (pp. 1163–1171). Red Hook, NY: Curran.

Schulz, A., Hinder, F., & Hammer, B. (2019). *Deepview: Visualizing classification boundaries of deep neural networks as scatter plots using discriminative dimensionality reduction.* arXiv:1909.09154.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., . . . Raffel, C. (2020). *Fixmatch: Simplifying semi-supervised learning with consistency and confidence.* arXiv:2001.07685.

Szubert, B., Cole, J. E., Monaco, C., & Drozdov, I. (2019). Structure-preserving visualisation of high dimensional single-cell datasets. *Scientific Reports*, *9*(1), 1–10. https://doi.org/10.1038/s41598-019-45301-0

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016). Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 287–297). New York: ACM. https://doi.org/10.1145/2872427.2883041

van der Maaten, L. (2009). Learning a parametric embedding by preserving local structure. In *Proceedings on the 12th International Conference on Artificial Intelligence and Statistics* (pp. 384–391).

van der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, *15*(1), 3221–3245.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

Venna, J., & Kaski, S. (2006). Local multidimensional scaling. *Neural Networks*, *19*(6–7), 889–899.

White, T. (2016). *Sampling generative networks.* arXiv:1609.04468.

Received April 1, 2021; accepted June 1, 2021.