Improving Out-of-sample Embedding in UMAP

Mohammad Tariqul Islam, *Student Member, IEEE,* and Jason W. Fleischer, *Fellow, Optica,*

Abstract—Neighbor embedding algorithms reveal correlations in high-dimensional data by constructing an equivalent graph representation in a lower-dimensional space. An increasingly popular algorithm is Uniform Manifold Learning and Projection (UMAP), which uses algebraic topology to map distances between the two spaces. While it works well on many types of data sets, UMAP has trouble adding new out-of-sample points to a pre-existing mapping. In particular, UMAP often places new points in the periphery of the found clusters, rather than the interior with their correlated neighbors. Here, we overcome this "repulsion effect" by optimizing pairwise interactions within the original k-nearest-neighbor graph. Moreover, we show that parameterizing UMAP obtains better embeddings than non-parametric algorithms, particularly as the data get more complex (e.g. medical images). We also show that the repulsion effect is naturally mitigated when parameterized UMAP is employed to embed the data. We characterize different UMAP approaches using a range of metrics, including trustworthiness, nearest neighbor classifier error, accumulation count, and analysis of the attractive and repulsive forces in the embeddings.

Index Terms—neighbor embedding, UMAP, neural network, parameterization, clustering

1 Introduction

Neighbor embedding algorithms are unsupervised algorithms that identify groups of related data. They are generally regarded as nonlinear dimensionality reduction methods. The methods define a pairwise metric between points in the high-dimensional space that is used to make a similar graph in a lower-dimensional space. Iterative neighborhood embedding methods, like t-distributed stochastic neighbor embedding (t-SNE) [1] and the recently introduced uniform manifold approximation and projection (UMAP) [2], have become standard methods due to their ability to scale up to millions of data points. In this section, we give a brief description history of the neighbor embedding framework. Afterwards, we focus on our attention on UMAP, as it has many significant properties that are not present in the other methods.

1.1 Neighbor Embedding Framework

The first step in the neighbor embedding framework is to construct a high-dimensional graph [3]. For n-dimensional data points $\{\mathbf{x}_i\}$ where $\mathbf{x}_i \in \mathbb{R}^n$ and $i = 1, 2, 3, \ldots, N$, the high-dimensional weighted graph is given by a pairwise metric

$$p_{ij} = f_H(d_H(\mathbf{x}_i, \mathbf{x}_j)|\{\mathbf{x}_i\}) \tag{1}$$

where $d_H(\cdot, \cdot)$ is a distance metric between two points and $f_H(\cdot)$ is a function that describes the weighted relation between points.

The low-dimensional graph is formulated in a similar manner:

$$q_{ij} = f_L(d_L(\mathbf{x}_i, \mathbf{x}_j)|\{\mathbf{y}_i\}) \tag{2}$$

- *This work is supported by ...*
- *Mohammad Tariqul Islam and Jason W. Fleischer are with the Department of Electrical and Computer Engineering, Princeton University, NJ, 08544. E-mail: mtislam@princeton.edu.*
- *Jason W. Fleischer are with the Department of Electrical and Computer Engineering and Program in Applied and Computational Mathematics, Princeton University, NJ, 08544. E-mail: jasonf@princeton.edu.*

Manuscript received xxxx; revised xxxx.

where the subscript L refers to the lower d-dimensional embedding space and $\mathbf{y}_i \in \mathbb{R}^d$, with $i = 1, 2, 3, \ldots, N$.

Finally, a relation between the high-dimensional graph and the low- dimensional graph is established by optimizing a loss function

$$\mathcal{L} = \sum_{i,j} l(p_{ij}, q_{ij}). \tag{3}$$

This embedding framework has been used extensively in life sciences, including RNA sequences [4], [5], [6], [7], [8], SARS-CoV-2 [9], [10], [11], communication engineering [12], [13], and animal motifs [14]. They are also becoming increasingly useful in biomedical applications, including both clinical [15] and image [16] data analysis.

1.2 Related works

In the base algorithm, the correlations and corresponding graph are calculated explicitly, without a parameter representation. Classic methods in the neighbor embedding regime include Sammon mapping [17], Isomap [18], locally linear embedding [19], Laplacian Eigenmaps [20], and t-SNE [1]. Iterative algorithms, such as t-SNE, have excellent visualization performance but slow implementation speed.

The speed of the algorithm is largely attributed to two factors: 1) the initial computation of the dense high-dimensional graph and 2) the normalization operation required in every optimization step. Several methods have been proposed to accelerate the calculations. In the graph construction step, Tang et al. [21] used random projection trees and neighbor exploration to construct an approximate k-NN graph. In the optimization step, Maaten [22] and Yang et al. [23] used restricted sampling via the Burns-Hut approximation [24], Mikolov et al. introduced negative sampling [25], and Linderman et al. [26] implemented a faster Fourier transform for interpolation. Recently, McInnes et al. skipped the normalization step altogether and formulated the low-dimensional embedding by pairwise interactions alone [2]. The resulting algorithm - UMAP - uses a loss

function that depends explicitly on attractive and repulsive forces between points. The attractive forces ensure that local neighborhoods are preserved, while the repulsive forces ensure that points that are originally far apart stay far apart. Recently, it was shown that a balance of these forces explains the optimization of UMAP [27]. Recently, it is demonstrated that t-SNE and UMAP are samples from a spectrum of neighbor embeddings [28], [29]. Other recent trends in the literature include methods for extending the embedding for incremental dataset by progressively updating the k-nearest neighbor (k-NN) graph [30], by reconstructing cluster centers and preserving consistensty of embedding [31], and using a seeding dataset for manifold alignment [32].

Force balance becomes problematic when new points are considered, i.e. out-of-sample points that weren't included in the original embedding. Indeed, the formulation in Section 1.1 requires all the weights to be computed again from scratch when new data is added. However, if there is a parametric transformation from high-dimensional data to low-dimensional data, such reconstruction can be avoided. Unfortunately, parametric embeddings are hard to obtain, due to the costly normalization required during each optimization step. One remedy is to divide the whole dataset into small subsets of data and optimized all subsets on the same parameters [33], [34]. Other approaches have been considered farther down the network pipeline, e.g. as a learned regularizer in the bottleneck of an autoencoder [35], directly in the loss function itself [36], or by computing weight functions in the intermediate layers of the neural network [37]. However, their effect on the visualization performance of out-of-sample embeddings has not been discussed.

1.3 Contribution

As comparisons with other neighbor embedding algorithms are done elsewhere, e.g. [36], our focus is on understanding and mitigating out-of-sample embedding errors in UMAP. The contributions of the paper are given below:

- We show that during embedding out-of-sample test points in UMAP, repulsive forces can result in test points accumulating in the periphery of clusters.
- We analyze placements of out-of-sample points through attractive and repulsive forces. We introduce the attractive force ratio (AFR) and the repulsive force ratio (RFR) as metrics that characterize the force balance in an embedding.
- We develop several deep neural network techniques that learn the embeddings, enabling parameterized versions of UMAP. We show that parameterized UMAP mitigates the adverse effects of repulsion and gives better embeddings.

To the best of our knowledge, this is the first paper that thoroughly analyzes the test embedding performance of the UMAP algorithm, identifies the core issue with test embeddings, and provides solutions to it. We also uncover the conditions in which parameterized embeddings work best, viz. fields with complex data that are generated continuously, such as those in biomedical and healthcare settings.

2 Uniform Manifold Approximation and Projection (UMAP)

In this section, we review the UMAP algorithm [2]. We then focus on embedding out-of-sample points, with an emphasis on the competition between attractive and repulsive forces.

2.1 Embedding Algorithm

In UMAP, a high-dimensional weighted relation between points is constructed using a k-NN graph. For each data point $\mathbf{x}_i$, the relation is given by

$$p_{i|j} = \begin{cases} \exp\left(-\frac{d_H(\mathbf{x}_i,\mathbf{x}_j)-\rho_i}{\sigma_i}\right) & \text{if } x_j \in \text{KNN}(\mathbf{x}_i, k) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\text{KNN}(\mathbf{x}_i, k)$ is the set of k-nearest neighbors of the point $\mathbf{x}_i$, $\rho_i = \min_{\mathbf{x}_j \in \text{KNN}(\mathbf{x}_i,k)} d_H(\mathbf{x}_i, \mathbf{x}_j)$ and σ_i is a scaling parameter set such that $\sum_j p_{i|j} = \log_2(k)$. The parameter ρ_i ensures that the point $\mathbf{x}_i$ has strong connectivity ($p_{i|j} = 1$) to at least one of the nearest neighbors, and the scaling parameter σ_i ensures the uniform manifold approximation. The adjacency matrix of the high-dimensional undirected weighted graph is obtained by (the probabilistic t-conorm),

$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j} \times p_{j|i} \tag{5}$$

The low-dimensional weight is given by a differentiable function

$$q_{ij} = \frac{1}{1 + a(||\mathbf{y}_i - \mathbf{y}_j||_2^2)^b} \tag{6}$$

where the parameters a and b determine how crowded the low-dimensional points become after embedding. These two parameters are chosen by fitting q_{ij} to

$$\Psi(\mathbf{y}_i, \mathbf{y}_j) = \begin{cases} 1 & \text{if } ||\mathbf{y}_i - \mathbf{y}_j||_2 < m_d \\ \exp(-(||\mathbf{y}_i - \mathbf{y}_j||_2 - m_d)) & \text{otherwise} \end{cases} \tag{7}$$

where m_d is a user-defined parameter that regulates the minimum distance between two low-dimensional points.

In order to obtain the final low-dimensional embedding, the UMAP algorithm aims to minimize the cross-entropy loss function

$$\mathcal{L} = \sum_{i,j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right) \tag{8}$$

from an initialization of the embedding $\{\mathbf{y_i}\}$. The first term of the loss function provides attractive forces and the second term provides repulsive forces. However, computing attractive-repulsive forces for all pairs of points is costly. Thus, UMAP takes a negative sampling approach [21], [25]. In the optimization cycle for each edge with p_{ij} in the k-NN graph, termed positive edge, a number of random edges are sampled, termed as negative edges. The attractive forces are minimized for the positive edges and the repulsive forces are minimized for negative edges. For the negative edges, the term $(1 - p_{ij})$ is assumed to be 1 (This assumption is often justified as most of the p_{ij} entries are zeros). In order to obtain a fast convergence, in each iteration, the method samples one positive edge with probability p_{ij} and applies the attractive force. After that, it randomly samples n_s negative edges and applies repulsive forces to each of the negative samples.

The parameter n_s, known as the negative sampling rate, is typically set to 5. A detailed treatment of the loss function (8) of UMAP is described in [27].

2.2 Embedding out-of-sample points in an existing embedding

One of the major drawbacks of the UMAP algorithm (as well as any non-parametric neighbor embedding algorithm) is that the methods are not online; if there is a need to embed out-of-sample or test points, the algorithm must recompute p_{ij} and q_{ij} from scratch. However, there are approximations that can place an out-of-sample point within an existing embedding. The simplest one optimizes the low-dimensional embedding of the new point alone while keeping the existing embedding intact [14], [38]. A more complex method, using a k-NN graph of the sample with respect to training dataset, appears in the UMAP software itself [39] but, to our knowledge, no description of the method has been given. For completeness, we give such a description now.

Let $\mathbf{u} \in \mathbb{R}^n$ be the out-of-sample test point, and $\mathbf{v} \in \mathbb{R}^d$ be its UMAP embedding. The high-dimensional weighted graph for this point is given by modifying (4) with the vector u:

$$p_{\mathbf{u}|j} = \begin{cases} \exp\left(-\frac{d(\mathbf{u},\mathbf{x}_j)-\rho_{\mathbf{u}}}{\sigma_{\mathbf{u}}}\right) & \text{if } \mathbf{x}_j \in \text{KNN}(\mathbf{u},k) \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

This is normalized to $p_{\mathbf{u}j}$ by

$$p_{\mathbf{u}j} = \frac{p_{\mathbf{u}|j}}{\sum_j p_{\mathbf{u}|j}}. \tag{10}$$

We next define the low-dimensional weight as

$$q_{\mathbf{v}j} = \frac{1}{1 + a(||\mathbf{v} - \mathbf{y}_j||_2^2)^b}. \tag{11}$$

From an initialization of $\mathbf{v}$, we aim to minimize the cross-entropy loss function (8) with respect to $\mathbf{v}$:

$$\mathcal{L} = \sum_{i,j} p_{\mathbf{u}j} \log\left(\frac{p_{\mathbf{u}j}}{q_{\mathbf{v}j}}\right) + (1 - p_{\mathbf{u}j}) \log\left(\frac{1 - p_{\mathbf{u}j}}{1 - q_{\mathbf{v}j}}\right). \tag{12}$$

Similar to the embedding algorithm in Section 2.1, in each iteration the attractive and the repulsive forces are applied to a single positive edge and n_s negative edges.

2.3 Repulsion Effect

Two main problems arise during the embedding of out-of-sample or test points: 1) the lack of normalization while approximating the low-dimensional structure in Eq. (11) and 2) the stochastic nature of the optimization from the negative sampling rate parameter n_s. Since there is no normalization of the low-dimensional embedding, each update from optimizing Eq. (12) works as a freewheel, i.e. there is no mitigating factor that reduces overshooting of the out-of-sample point from a desired location. Secondly, as the negative sampling parameter n_s typically is larger than 1, the out-of-sample point undergoes more repulsion than attraction while being optimized. The update equation of an out-of-sample point v is given by a series of equations

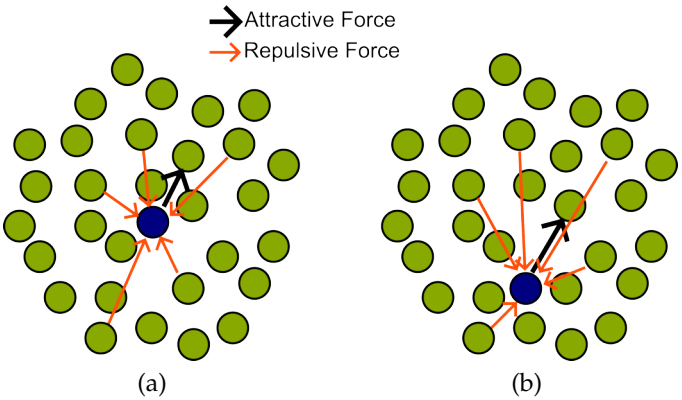


Fig. 1: Influence of attractive force from a neighbor and repulsive forces from negative samples on a out-of-sample test point (blue). (a) When the point is in the interior of the cluster the negative samples often surround the test point. Thus the overall forces balance each other. (b) However, when the point is near the periphery of the cluster, the negative samples often provide an overall directional force that may cause the test point to be pushed to the periphery of the cluster, resulting in what we call 'repulsion effect'.

$$v^{(t+a)} = v^{(t)} - \eta f_a(v^{(t)}, y^{(a)}), \tag{13}$$

$$v^{(t+a+1)} = v^{(t+a)} + \eta f_r(v^{(t+a)}, y^{(1)}), \tag{14}$$

$$\dots,$$

$$v^{(t+1)} = v^{(t+a+n_s)} = v^{(t+a+n_s-1)} + \eta f_r(v^{(t+a+n_s-1)}, y^{(n_s)}) \tag{15}$$

where t is the iteration number, $v^{(t+a)}$ is the update due to attractive forces from the neighbor $y^{(a)}$, $\{v^{(t+a+1)}, \dots, v^{(t+a+n_s)}\}$ are updates due to repulsive forces from the negative samples $\{y^{(1)}, \dots, y^{(n_s)}\}$, η is the learning rate, and f_a and f_r define the attractive and repulsive force, respectively.

The multiple repulsions are not a problem if the repulsive forces are distributed evenly, which prevents the overall effect from being directional (Fig. 1(a)). If most of the repulsive forces are in the same direction, however (e.g. near a boundary), then the resultant repulsive force pushes the point towards the periphery of the original embedding (Fig. 1(b)). For very strong repulsive forces, occurring e.g. when the number of nearest neighbors k is high, the test point may shoot itself out of the cluster boundary. We dub this phenomenon the "repulsion effect". On the other hand, if n_s is set to zero for out-of-sample points, then the attractive force will dominate and the point will manifest in the middle of the embedding, close to its nearest neighbor partner. Thus in order to obtain an optimal placement, the values k and n_s need to be tuned individually for each out-of-sample point, which is a time-consuming process. Moreover, the user typically does not know the best placement of the out-of-sample point, making manual tuning a futile effort.

3 PARAMETERIZING UMAP

In order to parameterize UMAP, we define, $y_i = f_\theta(x_i)$, where $\{\theta\}$ is a set, $f_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^d$ is a transformation from the high-dimensional input to the low-dimensional output of controlling parameters θ. We assume that the function $f_\theta(\cdot)$ is differentiable in order to optimize using gradient methods. There are many approaches to obtain a parameterized version of UMAP, e.g, optimizing the cross-entropy loss function (as in the original UMAP algorithm) or mean-squared error. In what follows, we denote the class of parameterized versions of UMAP as P-UMAP.

3.1 Minimizing Cross Entropy Error: P-UMAP-CE

Here, we formulate the embedding optimization as a 'learning UMAP from first principles' concept. First, we define the low-dimensional weights similar to (6)

$$q_{ij}^{\{\mathrm{P}\}} = \frac{1}{(1 + a(||f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)||_2^2)^b)}. \tag{16}$$

Then, we minimize the cross-entropy loss function

$$\mathcal{L}^{\{\mathrm{P}\}} = \sum_{i,j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}^{\{\mathrm{P}\}}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}^{\{\mathrm{P}\}}}\right) \tag{17}$$

with respect to θ to obtain a low-dimensional. This method resembles the formulation of [36].

3.2 Regression using mean squared error: P-UMAP-MSE

In this approach, a realization $\{y_i\}$ of the UMAP is obtained by running the UMAP algorithm to obtain a reference embedding $\mathbf{y}$. After that, a parameterized approximation is obtained simply by minimizing the mean-squared error

$$\mathcal{L}^{\{\mathrm{MSE}\}} = \sum_i ||f_\theta(\mathbf{x}_i) - \mathbf{y}_i||_2^2. \tag{18}$$

This is a straightforward, inexpensive approach.

3.3 Combination approach: P-UMAP-CEMSE

In this approach, we obtain a parameterized version of UMAP by minimizing both the cross-entropy error and the mean-squared error in the combined loss function

$$\mathcal{L}^{\{\mathrm{P2}\}} = \mathcal{L}^{\{\mathrm{P}\}} + \mathcal{L}^{\{\mathrm{MSE}\}}. \tag{19}$$

This joint function combines the quick learning of MSE error with the data-driven structure of CE.

4 EXPERIMENTS

Experiments are carried out to evaluate how out-of-sample points are placed in the embeddings. We first describe the experimental setup and the metrics used to evaluate the embeddings. We then describe the visulaization performance in different datasets. Finally, we characterize the placement of new points in terms of attractive and repulsive .

4.1 Experimental Setup

For parameterizing UMAP, we employed a seven-layer fully connected, feed-forward neural network. Each layer is defined by

$$f_l(\mathbf{x}) = \sigma(\gamma_l(\mathbf{W}_l\mathbf{x} + \mathbf{b_l})) \tag{20}$$

where l is the layer number, $\sigma(\cdot)$ is the sigmoid activation function, and $\mathbf{W}_l$, $\mathbf{b}_l$, and γ_l are learnable weights, biases and scaling parameters, respectively, θ. The number of output neurons of each of the layers are 500, 300, 200, 100, 100, 100, and d.

The Adam [40] algorithm has been used to optimize the network parameters for each loss method. The learning rate is set to 0.001 initially and then decreased successively by $1/10$th every five epochs. In order to optimize the network, we followed the same negative sampling principle of UMAP and used $n_s = 5$ for all networks. Depending on the dataset and learning mechanism, we changed the number of epochs in the network training. For UMAP-MSE and UMAP-CEMSE, the number of epochs is set to 20. Compared to UMAP-MSE and UMAP-CEMSE, UMAP-CE is a slow learner; thus, we used 40 epochs for UMAP-CE. Any changes from these settings are discussed in respective parts of discussion.

All of our experiments are coded in Python 3.6.9. We employed pyTorch software package [41] to run the deep learning experiments. UMAP embeddings were obtained using UMAP software [39] v0.4.6. Our own codes for different metrics are implemented using numpy [42], numba v0.51 [43] and scikit-learn [44].

4.2 Evaluation metrics

A single evaluation metric often fails to capture the diverse properties of an embedding. Therefore, we evaluate in multitude ways to assess the embeddings comprehensively. In a global context, we assess the embeddings based on two metrics: the trustworthiness metric (T_κ) [45] and the k-NN classifier error. To quantify the 'repulsion effect', we need to analyze the accumulation of points at the periphery of a cluster. We characterize it in two ways: 1) by counting the number of points that accumulate in the periphery (accumulation), and 2) by analyzing the balance of attractive and repulsive forces around the periphery in terms of force ratios. The metrics are described in detail below.

4.2.1 Trusworthiness

The trustworthiness metric measures whether the obtained local structure in low-dimensional data conforms with the structure of the high-dimensional data by

$$T_\kappa = 1 - \frac{2}{n\kappa(2n - 3\kappa - 1)} \sum_{i=1}^{n} \sum_{y_j \in \mathrm{KNN}(y_i,\kappa)} \max(0, (r(i,j) - \kappa)) \tag{21}$$

where $KNN(y_i, \kappa)$ is the k-NN graph in the low-dimensional space and $r(i,j)$ is the rank of x_j in the high-dimensional k-NN graph ($KNN(x_i, \kappa)$). The parameter $\kappa(< \frac{N}{2})$ refers to the number of nearest neighbors considered in the low dimension. Low (high) value of κ corresponds a smalle (large) local structure. We can also define κ as

the scale of the trustworthiness metric. To understand the structure preservation in the nearest neighbor sense for various sizes of local structure, we consider values of κ to 5, 30, and 100. Moreover, we consider trustworthiness of training data alone, which is traditional, and trustworthiness of combined training and test data to see where out-of-sample test points would be placed. Previous literature [36] computed trustworthiness by sampling 10,000 points from the embedding; here, we used all the available samples in each dataset to paint a more accurate picture.

4.2.2 k-NN Classifier

k-nearest neighbor (k-NN) [46] classifier is used to check the resulting clustering properties of the algorithms. We employ 1-NN and 5-NN classifiers. 1-NN classifier assigns the label of nearest neighbor to the test point. 5-NN classifier considers 5 nearest neighbors and assigns the majority of the labels to the test point.

4.2.3 Accumulation

Given a cluster $\mathcal{C}$ and the associated test points $\{\mathbf{v}_\mathcal{C}\}$, we define accumualtion ζ by

$$\zeta = |\{\mathbf{v}|\mathbf{v} \in \mathbf{v}_\mathcal{C}, \mathbf{v} \notin P\}| \tag{22}$$

where P represents a convex polytope within $\mathcal{C}$. To obtain the desired convex polytope, we first compute the convex hull of the cluster. Subsequently, we adjust the vertices of the convex hull by moving them towards the center, reducing their distance from the center by 5%. Further details can be found in Supplementary Section S.I.

4.2.4 Force Ratios

The forces are inherently d-dimensional vectors. In order to define the force ratios, however, we only work with scalars. The total attractive force on an embedded point $\mathbf{v}$ (and corresponding high-dimensional point $\mathbf{u}$) is given by

$$F_a(\mathbf{v}) = \Big|\Big|\sum_j p_{\mathbf{u}|j} f_a(\mathbf{v}, \mathbf{y}_j)\Big|\Big|, \quad \text{where} \quad \mathbf{x}_j \in \text{KNN}(\mathbf{u}, k) \tag{23}$$

where $f_a(\mathbf{v}, \mathbf{y}_j)$ is the pairwise attractive force between embedded points $\mathbf{v}$ and $\mathbf{y}_j$, and $p_{\mathbf{u}|j}$ is the weight of the point $\mathbf{u}$ in the high-dimensional graph. Similarly, the total repulsive force on the point $\mathbf{v}$ is given by

$$F_r(\mathbf{v}) = \Big|\Big|\sum_j p_{\mathbf{u}|j} f_r(\mathbf{v}, \mathbf{y}_j)\Big|\Big|, \quad \text{where} \quad \mathbf{x}_j \in \text{KNN}(\mathbf{u}, k) \tag{24}$$

where $f_r(\mathbf{v}, \mathbf{y}_j)$ is the pairwise repulsive force between $\mathbf{v}$ and $\mathbf{y}_j$.

With these averages, the force ratio is defined as ratio of forces of two sets of embedded points S_1 and S_2. The attractive force ratio (AFR) is given by the ratio of average attractive forces on points in S_1 and the points in S_2

$$\text{AFR}(S_1||S_2) = \frac{\sum_{\mathbf{v}\in S_1} F_a(\mathbf{v})}{\sum_{\mathbf{z}\in S_2} F_a(\mathbf{z})}. \tag{25}$$

In the ideal case, the AFR should be close to 1.0 indicating the balance of force around the neighborhood of the points of the sets. As a result, it can be used as a metric of force balance (For simplicity, one can define new metrics such as $1 - \text{AFR}$ or $|1 - \text{AFR}|$, so that the balance of force occurs at 0). It is to be noted that AFR is not symmetric function, thus typically $\text{AFR}(S_1||S_2)$ and $\text{AFR}(S_2||S_1)$ are not equal.

In a similar fashion, the repulsive force ratio (RFR) is defined

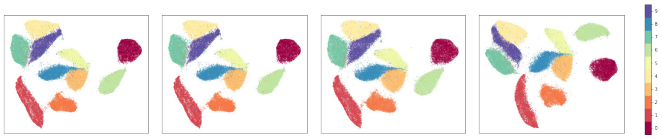
$$\text{RFR}(S_1||S_2) = \frac{\sum_{\mathbf{v}\in S_1} F_r(\mathbf{v})}{\sum_{\mathbf{z}\in S_2} F_r(\mathbf{z})}. \tag{26}$$

As previously, balance is indicated at a value of 1.0. Similarly to AFR, $\text{RFR}(S_1||S_2)$ and $\text{RFR}(S_2||S_1)$ are not equal.

The force ratios in Eq. (25) and (26) are defined abstractly and thus can be implemented for any of the algorithms under the neighbor embedding framework. For the UMAP algorithm, the attractive force f_a and the repulsive force f_r are obtained by diferentiating the atttractive and repulsive terms of Eq. 8, respectively:

$$f_a(\mathbf{v}, \mathbf{y}) = \frac{2ab(||\mathbf{v}-\mathbf{y}||_2^2)^{b-1}}{1 + a*(||\mathbf{v}-\mathbf{y}||_2^2)^b}(\mathbf{v}-\mathbf{y}), \tag{27}$$

$$f_r(\mathbf{v}, \mathbf{y}) = \frac{2b}{||\mathbf{v}-\mathbf{y}||_2^2(1 + a(||\mathbf{v}-\mathbf{y}||_2^2)^b)}(\mathbf{v}-\mathbf{y}). \tag{28}$$



(a) (b) (c) (d)

Fig. 2: Two-dimensional embedding of the MNIST dataset using (a) UMAP, (b) P-UMAP-MSE, (c) P-UMAP-CEMSE, and (d) P-UMAP-CE. Note that, (b) and (c) use (a) as the reference embedding. The figures (b)-(d) shows that all the parameterized methods obtain comparable embeddings to the original UMAP method in (a).

4.3 Comparing Visualisations

In this section, we compare visulaization performance of different approaches in three different datasets: MNIST, chest x-rays, and clinical data from emergency departments. In order to evaluate the embeddings, we employed trustworthiness metric (T_κ) [45] and the k-NN classifier [46] error. The value of κ determines the size of the local neighborhood around which the embedding quality is measured. We compare our results to the original UMAP algorithm and by varying the nearest neighbor parameter k and the negative sampling parameter n_s.

4.3.1 MNIST

As a commonly used benchmark for evaluating neighbor embedding methods, we start with the standard MNIST dataset of handwritten digit images [47]. For embedding MNIST, the nearest-neighbor parameter k is set to 30 and the minimum-distance parameter m_d is set to 0.25. For UMAP, the training embedding is obtained using the default $n_s = 5$. After that, we examined values of $n_s = 3$ and 1 to see the effect of repulsive forces. We also obtained embedding for combined training and test data (UMAP (Train+Test)), i.e. the entire dataset, to see how UMAP would "naturally" place the out-of-sample

TABLE 1: Results on MNIST dataset in terms of Turstworthiness and k-NN classifier at different scales.

	Trustworthiness (Train)			Trustworthiness (Train+Test)			k-NN classifier		Accumulation, ζ
	T_5	T_{30}	T_{100}	T_5	T_{30}	T_{100}	1-NN Error	5-NN Error	Label 1
UMAP ($n_s = 5$)	0.9523	0.9518	0.9502	0.9523	0.9519	0.9505	7.41%	4.74%	104
UMAP ($n_s = 3$)	–	–	–	0.9523	0.9519	0.9517	7.41%	4.80%	75
UMAP ($n_s = 1$)	–	–	–	0.9523	0.9518	0.9505	7.13%	4.74%	48
UMAP (Train+Test)	0.9537	0.9531	0.9517	0.9542	0.9533	0.9521	**6.14%**	**3.64%**	49
P-UMAP-MSE	0.9523	0.9517	0.9491	0.9467	0.9467	0.9456	10.24%	7.23%	53
P-UMAP-CEMSE	0.9550	0.9544	0.9522	0.9538	0.9534	0.9523	6.67%	4.19%	46
P-UMAP-CE	**0.9555**	**0.9553**	**0.9535**	**0.9547**	**0.9546**	**0.9538**	8.20%	5.02%	51

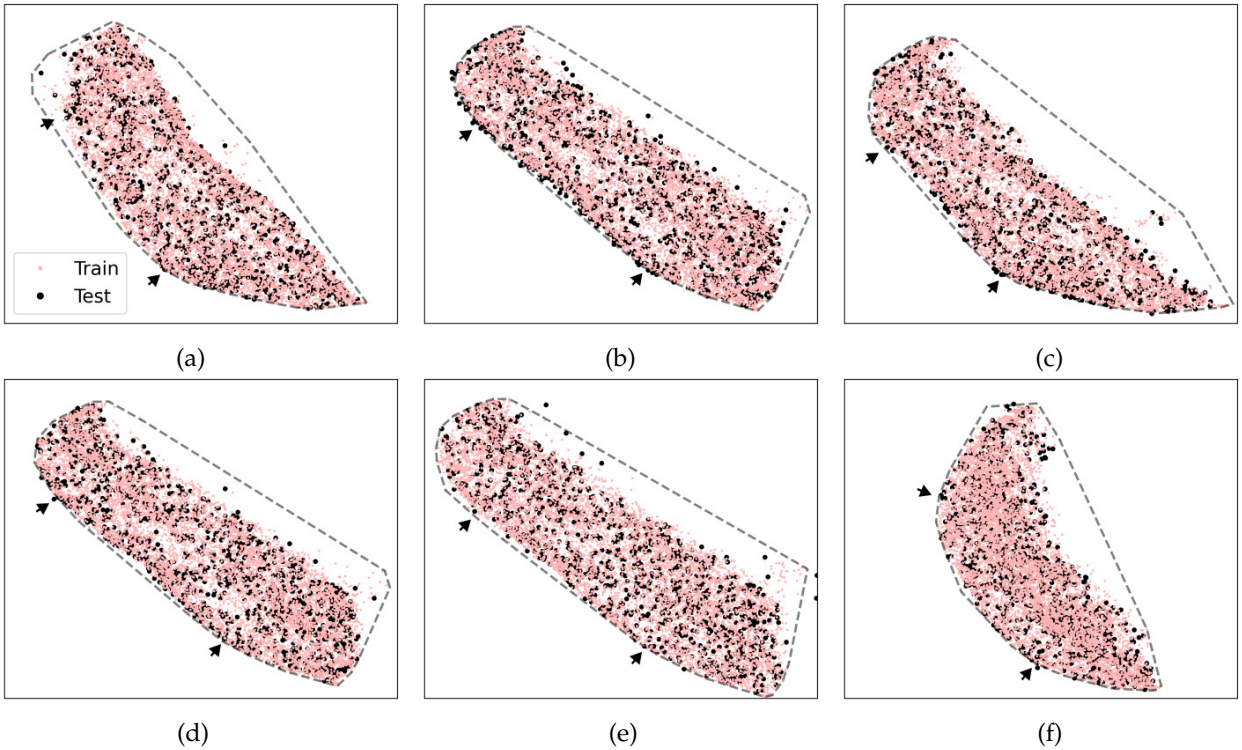


Fig. 3: Two-dimensional embedding of MNIST data, zoomed to show the cluster labeled 1. The convex hull of this cluster identified by HDBSCAN is shown using grey dashed line. (a) UMAP algorithm with $k = 30$ with both training data and test data embedded together ($\zeta = 49$). (b,c) UMAP algorithm for training data followed by the test data considered as out-of-sample data using method in Section 2.2 with (b) $k = 30$ ($\zeta = 104$) and (c) $k = 15$ ($\zeta = 105$). (d) UMAP algorithm as in (b) but n_s is set to 3 for test embedding ($\zeta = 75$). (e) P-UMAP-CEMSE ($\zeta = 46$), and (f) P-UMAP-CE ($\zeta = 51$) for the same settings as (b). Both (b) and (c) show accumulation of test points (in black) at the periphery of the cluster (indicated using black arrow) due to 'repulsion effect'. This accumulation is not present in the rest.

TABLE 2: Time required to embed each out-of-sample point of MNIST data for $k = 30$ for UMAP algorithm, and a trained neural network.

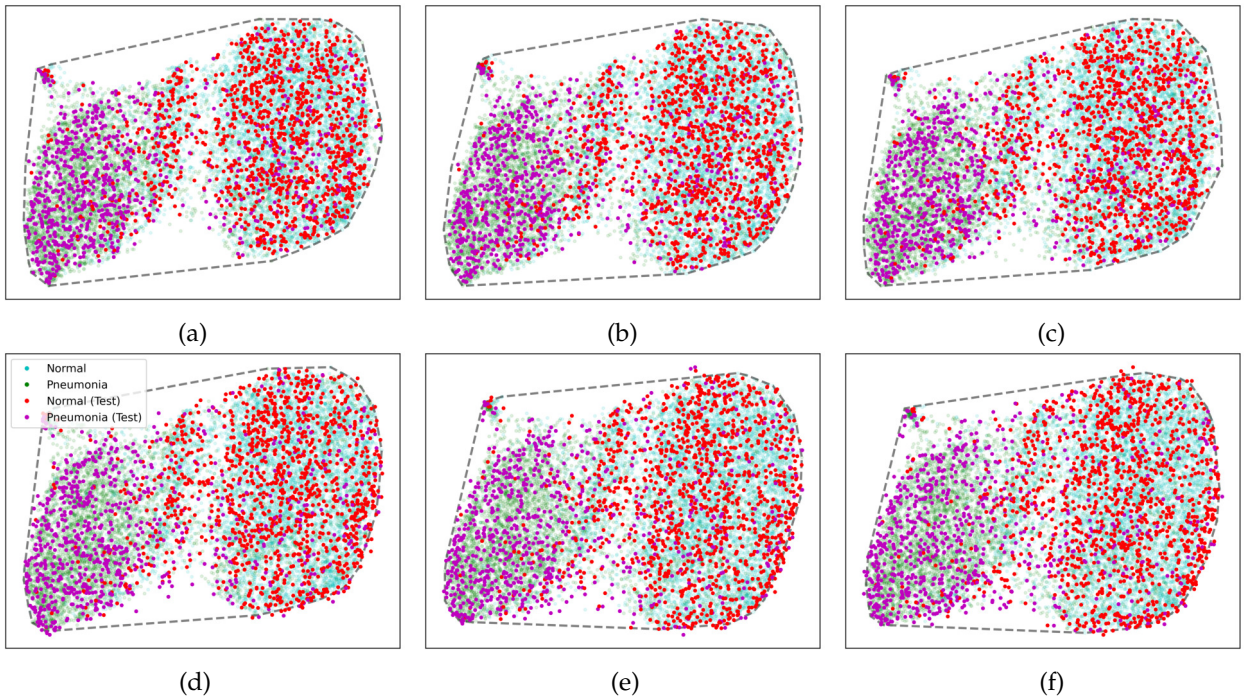
Method	Sequential Time (ms)	Parallet Time (ms)
UMAP	489.63	5.79
Network	0.53	0.08

test points. Numerical results for these and the parameterized UMAP embeddings are shown in Fig. 2 and summarized in Table 1. The embeddings are quite similar visually with the trustworthiness values for varying n_s close to each other, as MNIST already has a clearly defined manifold. Nevertheless, we can see that the overall embedding in terms of trustworthiness is better for T_{100} when $n_s = 3$ is used, which indicates a better preservation of larger structure. However, the trustworthiness values are even higher when the training and test data are embedded together (UMAP (Train+Test)). The gap between this and UMAP implementation illustrates the misplacements due to 'repulsion effect'. For parameterized embeddings, P-UMAP-MSE cannot overcome the drawbacks of UMAP, whereas the P-UMAP-CEMSE and P-UMAP-CE embeddings provide higher trustworthiness. In terms of the k-NN classifier error, the embedding of UMAP (Train+Test) shows the best performance. Interestingly, P-UMAP-CEMSE approach provides the 2nd-best performance (We will see later that the CE loss alone provides the best result as the datasets become more complex).

The previous metrics reduce the embeddigns to a single number and do not give any indicationn of the distribution of points. We now turn our attention to the repulsion effect of the UMAP algorithm. Whereas the repulsion effect is

TABLE 3: Results on chest x-ray embedding in terms of Turstworthiness and k-NN classifier at different scales.

	Trustworthiness (Train)			Trustworthiness (Train+Test)			k-NN classifier		Accumulation, ζ
	T_5	T_{30}	T_{100}	T_5	T_{30}	T_{100}	1-NN Error	5-NN Error	
UMAP $_{(n_s=5)}$	0.8345	0.8332	0.8315	0.8335	0.8326	0.8314	23.85%	19.05%	245
UMAP $_{(n_s=3)}$	–	–	–	0.8340	0.8330	0.8315	25.05%	17.85%	79
UMAP $_{(n_s=1)}$	–	–	–	0.8321	0.8312	0.8301	22.89%	17.50%	30
UMAP $_{\text{(Train+Test)}}$	0.8328	0.8324	0.8308	0.8340	0.8330	0.8315	23.90%	17.85%	76
P-UMAP-MSE	0.8345	0.8338	0.8315	0.8325	0.8323	0.8305	23.30%	17.90%	63
P-UMAP-CEMSE	0.8397	0.8410	0.8411	0.8379	0.8393	0.8398	23.15%	17.00%	70
P-UMAP-CE	**0.8430**	**0.8430**	**0.8429**	**0.8400**	**0.8403**	**0.8409**	**22.30%**	**16.15%**	78



(a) (b) (c) (d) (e) (f)

Fig. 4: X-ray features embedded using UMAP. Training points are shown using transparent colors in the background, and test points are shown using solid colors. The convex hull of the embedding of the training data is shown using gray dotted line. Top row: training and test data embedded together using (a) $k = 15$ ($\zeta = 82$), (b) $k = 30$ ($\zeta = 76$) and (c) $k = 50$ ($\zeta = 62$). Bottom row: Embedded training followed by out-of-sample test points using (d) $k = 15$ ($\zeta = 202$), (e) $k = 30$ ($\zeta = 245$) and (f) $k = 50$ ($\zeta = 243$) following rule-based algorithm in Section 2.2. (a-c) establishes baseline of embeddings for increasing k as both training and test points are embedded together and test embeddings are mostly inside the convex hull. (d-f) show that when test points are embedded afterwards, they are placed around the periphery of the mapping and crossing the convex hull due to 'repulsion effect'; as k is increased, the repulsion effect increases, causing the number of test points in the periphery to increase.

not severe or sometimes noticeable in the MNIST dataset, it is present. To demonstrate the 'repulsion effect', we first identify the cluster primarily consisting of Label 1 using HDBSCAN [48] and then compute accumulation ζ (as displayed in Table 1). The baseline value of $\zeta = 49$ is obtained for UMAP $_{\text{(Train+Test)}}$. For UMAP $_{(n_s=5)}$, $\zeta = 104$, which is more than double of the nominal value, indicating that the repulsive forces have pushed more test points to the periphery of the cluster. Similarly, for $n_s = 3$, accumulation is higher than nominal ($\zeta = 75 > 49$), whereas for $n_s = 1$ matches that with the nominal one. On the other hand, all the parameterized UMAP implementation are relatively close to the nominal value. Next, we visually analyze some of these clusters in Fig. 3. Fig. 3(c) is a UMAP for the nearest neighbor parameter $k = 15$, whereas the rest are for $k = 30$. The convex hull of the cluster is shown using a grey dashed line, and points of interest are shown using black arrows. It should be noted that the clusters obtained from the UMAP algorithm are not convex. However, in this example, the convex hull effectively demonstrates the repulsion effect in the UMAP embeddings. For reference, when both the training and test data are used to obtain the UMAP embedding (Fig. 3(a)), the test points (black dots) are embedded throughout the cluster of training points (red dots) with $\zeta = 49$. When the test points are added as new data after the initial training, many test points accumulate at the boundary of the cluster (Figures 3(b,c) for $k = 30$ ($\zeta = 104$) and 15 (105) respectively, showing that the repulsion effect does not necessarily depend on the value of k. The accumulation is somewhat decreased in the UMAP with negative sampling parameter n_s set to 3

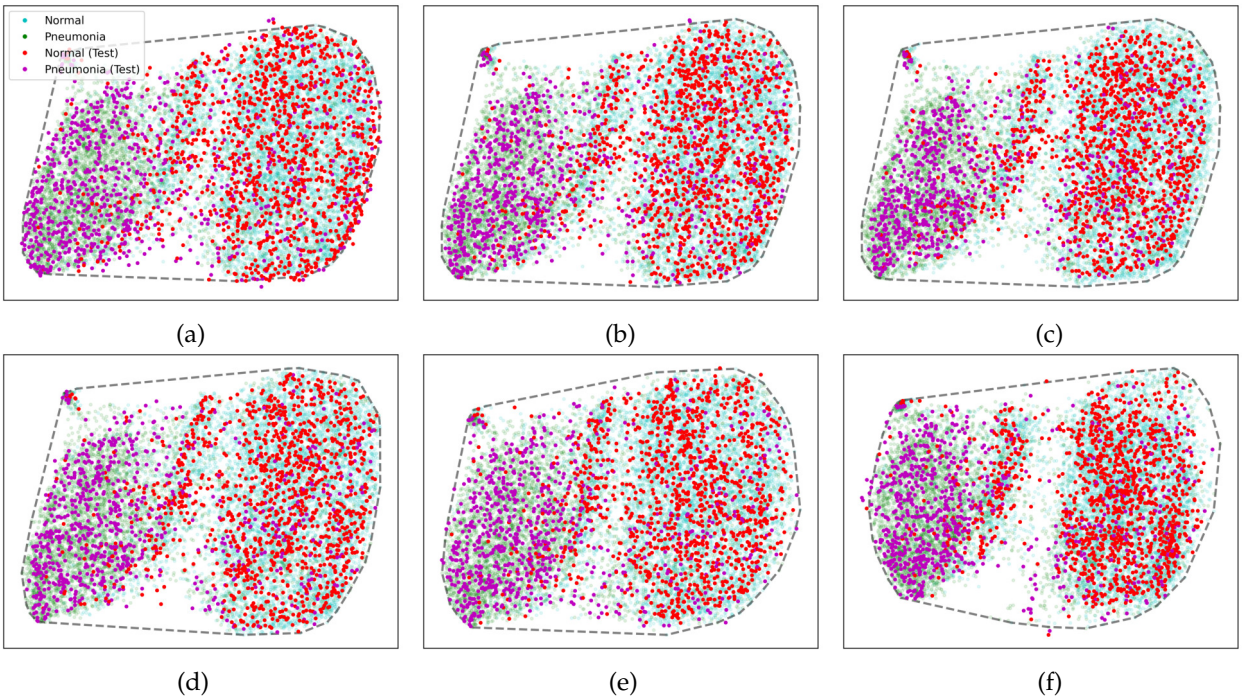


Fig. 5: Embeddings for x-ray data for different algorithms for $k = 30$. Training points are shown using transparent colors in the background, and test points are shown using solid colors. The convex hull of the training points is shown using a gray dotted line. Top row: UMAP for test points with n_s set to (a) 5 ($\zeta = 245$), (b) 3 ($\zeta = 79$), and (c) 1 ($\zeta = 30$). Bottom row: parametized embeddings using (d) P-UMAP-MSE ($\zeta = 63$), (e) P-UMAP-CEMSE ($\zeta = 70$), and (f) P-UMAP-CE ($\zeta = 78$). The accumulation of the points at the periphery is removed for the modified methods as seen in (b-f) and quantified in Table 3.

for the test points (Fig. 3(d), $\zeta = 75$). The accumulation is similar to baseline UMAP $_{\text{(Train+test)}}$ ($\zeta = 49$) for P-UMAP-CEMSE (Fig. 3(e), $\zeta = 46$) and P-UMAP-CE (Fig. 3(f), $\zeta = 51$).

Finally, we look at the time required to embed each out-of-sample test point. We computed the running time using 100 test samples on a CPU. (AMD Ryzen Threadripper 2950X 16-Core Processor with 128 GB of RAM). Table 2 summarizes the results for both sequential and parallel embeddings. In both cases, the neural network approach is faster by several orders of magnitude than the conventional, rule-based out-of-sample embedding in UMAP.

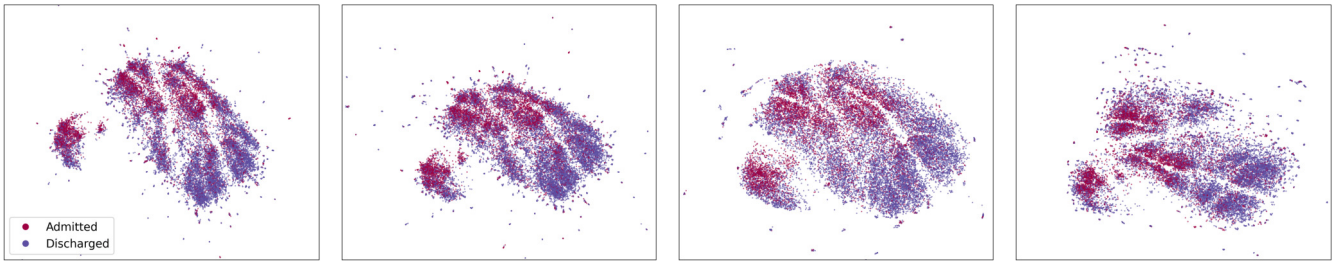
4.3.2 *Chest x-ray Image Embedding*

Unsupervised embedding of chest x-ray images as a method for exploratory analysis of diseases was first discussed in [16]. Here, we describe the embedding of chest x-rays with normal vs. pneumonia representations. From the RSNA Pneumonia Detection Dataset [49], we collected a total of 13389 chest x-rays: the training set consists of 6775 normal x-rays and 4614 pneumonia x-rays, while the test set consists of 1191 normal x-rays and 809 pneumonia x-rays. Because x-ray images have large variations, e.g. x-ray power and orientation during image capture as well as the condition of the subject, this dataset does not have a well-defined manifold like the images in MNIST. Thus, we used a pre-processing step to obtain a manifold by extracting features from the x-ray images. More specifically, we used a DenseNet-121 [50] (a deep neural network trained on Imagenet [51]) to obtain a set of 1024 characteristic features. As shown in Fig. 4, the resulting UMAP embedding formed weakly separable clusters of normal vs. pneumonia patients. The convex hull of the training embedding is shown using a gray dotted line.

This dataset shows a more obvious instance of the repulsion effect than MNIST. The baseline embeddings are established by embedding both the training and test data together (Fig. 4(a), (b), and (c) for $k = 30$, 50, and 70, respectively). Here, the accumulation decreases as k is increased. The embeddings when test points are embedded after training are shown in Fig. 4 (d), (e), and (f) for $k = 15$, $k = 30$ and $k = 50$, respectively. The accumulation is more than three times for these cases than the baseline ones. As k is increased from 15 to 30, the accumulation increases.

Fig. 5 shows different embeddings for $k = 30$. We used a negative sampling rate of $n_s = 5$ for all the training cases, but for testing purposes we also considered $n_s = 3$ and $n_s = 1$ (Figures 5, top row). As n_s is decreased the accumulation of points at border decreases. When $n_s = 3$, $\zeta = 79$ a value closer to the baseline (78 for UMAP $_{\text{(Train+Test)}}$), but a distinct avoidance of the border is apparent when $n_s = 1$ ($\zeta = 30$, again implicating repulsion as there is no repulsion effect when only one negative edge is considered). Similarly, the parameterized versions of UMAP do not require negative sampling after training; and there is no sign of any repulsion effect in their embeddings either (Figures 5, bottom row).

Table 3 summarizes the trustworthiness metric, performance of the k-NN classifier and accumulation. P-UMAP-CE gives the highest scores, with the performance of P-UMAP-

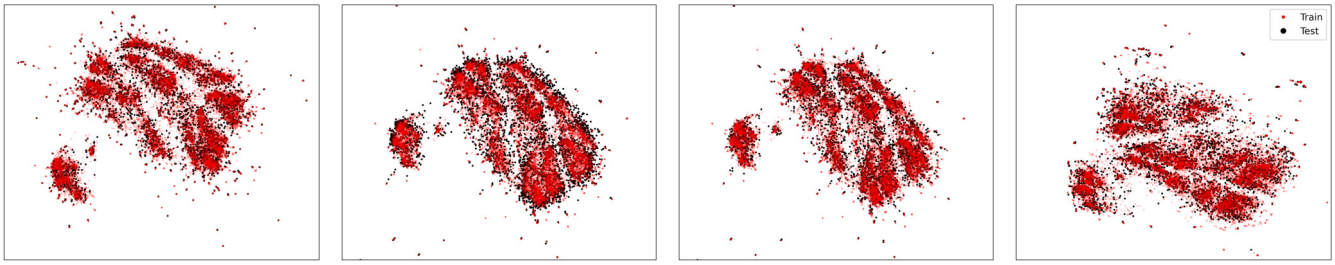


(a) (b) (c) (d)

Fig. 6: Embedding of patient features with 'shortness of breath' as the chief complaint. (a) UMAP, (b) P-UMAP-MSE, (c) P-UMAP-CEMSE, (d) P-UMAP-CE.

TABLE 4: Results on clinical dataset with 'shortness of breath' as the chief complaint in terms of Turstworthiness and k-NN classifier at different scales.

	Trustworthiness (Train)			Trustworthiness (Train+Test)			k-NN classifier	
	T_5	T_{30}	T_{100}	T_5	T_{30}	T_{100}	1-NN Error	5-NN Error
UMAP $_{(n_s = 5)}$	0.8229	0.7948	0.7819	0.8159	0.7942	0.7817	35.40%	31.45%
UMAP $_{(n_s = 3)}$	–	–	–	0.8179	0.7958	0.7828	35.36%	30.60%
UMAP $_{(n_s = 1)}$	–	–	–	0.8166	0.7944	0.7823	35.96%	30.79%
UMAP $_{\text{(Train+Test)}}$	0.8283	0.7975	0.7829	0.8303	0.8004	0.7853	35.21%	30.30%
P-UMAP-MSE	0.8119	0.7858	0.7741	0.7997	0.7803	0.7693	35.75%	30.95%
P-UMAP-CEMSE	0.8121	0.7998	0.7887	0.8070	0.7982	0.7870	34.90%	30.16%
P-UMAP-CE	**0.8431**	**0.8119**	**0.7994**	**0.8342**	**0.8098**	**0.7982**	**34.70%**	**29.75%**



(a) (b) (c) (d)

Fig. 7: Repulsion effect in the clinical dataset for patients with shortness of breath. (a) UMAP when training and test data are embedded together. (b-d) Embedded using training data followed by test data as out-of-sample points using (b) original UMAP embedding with $n_s = 5$, (c) UMAP embedding with $n_s = 3$, and (d) P-UMAP-CE. The repulsion effect leading to the accumulation of points can be observed in (b) through the presence of a noticeable dark shadow at the peripheries, which is absent in both (c) and (d).

CEMSE a close second. For the non-parametric UMAP algorithms, the trustworthiness on the complete dataset (Train+Test) is highest when $n_s = 3$, a lower value than that used in the original construction. However, this value cannot be too low, as with $n_s = 1$, the trustworthiness decreases. The accumulation is computed for the whole embeddings. The baseline value of ζ is 76 (for UMAP $_{\text{(Train+Test)}}$). For UMAP $_{(n_s = 5)}$, $\zeta = 245$ which is three times the nominal value. UMAP $_{(n_s = 3)}$ and P-UMAP-CE obtains values closer to the nominal one.

4.3.3 *Clinical Data*

Here, we examine the dataset compiled by Hong et al. [15] to automate hospital admissions in emergency departments. The data were collected from three departments from a period of March 2013 to July 2017 and consists of patient history, demographics, chief complaints, and vital signs. It includes patient triage information of 190,000 patients and a total of 560,486 hospital visits. The data is primarily labeled by a feature named 'disposition', indicating whether a patient was admitted into the hospital or discharged. In addition, 971 triage and demographic features were collected which include both categorical and numerical data. Here, we employ the method of patient phenotyping used by Hurley et al. [52] and consider 'Shortness of Breath' as the chief complaint. (Analysis of 'Abdominal Pain' is provided in Supplementary Section S.III.) The 'disposition' and 'emergency severity index' features were discarded, as they are decision variables, while the categorical data were one-hot encoded (missing data were mean-imputed). After that, the data was divided into train and test sets with an 80% and 20% split, respectively. All embeddings were obtained by setting the k-NN parameter k to 30. For the UMAP embedding, we set the minimum distance parameters m_d to 0, 0.0001, 0.001, 0.01, 0.1, 0.5, 0.8, 1.0 and reported

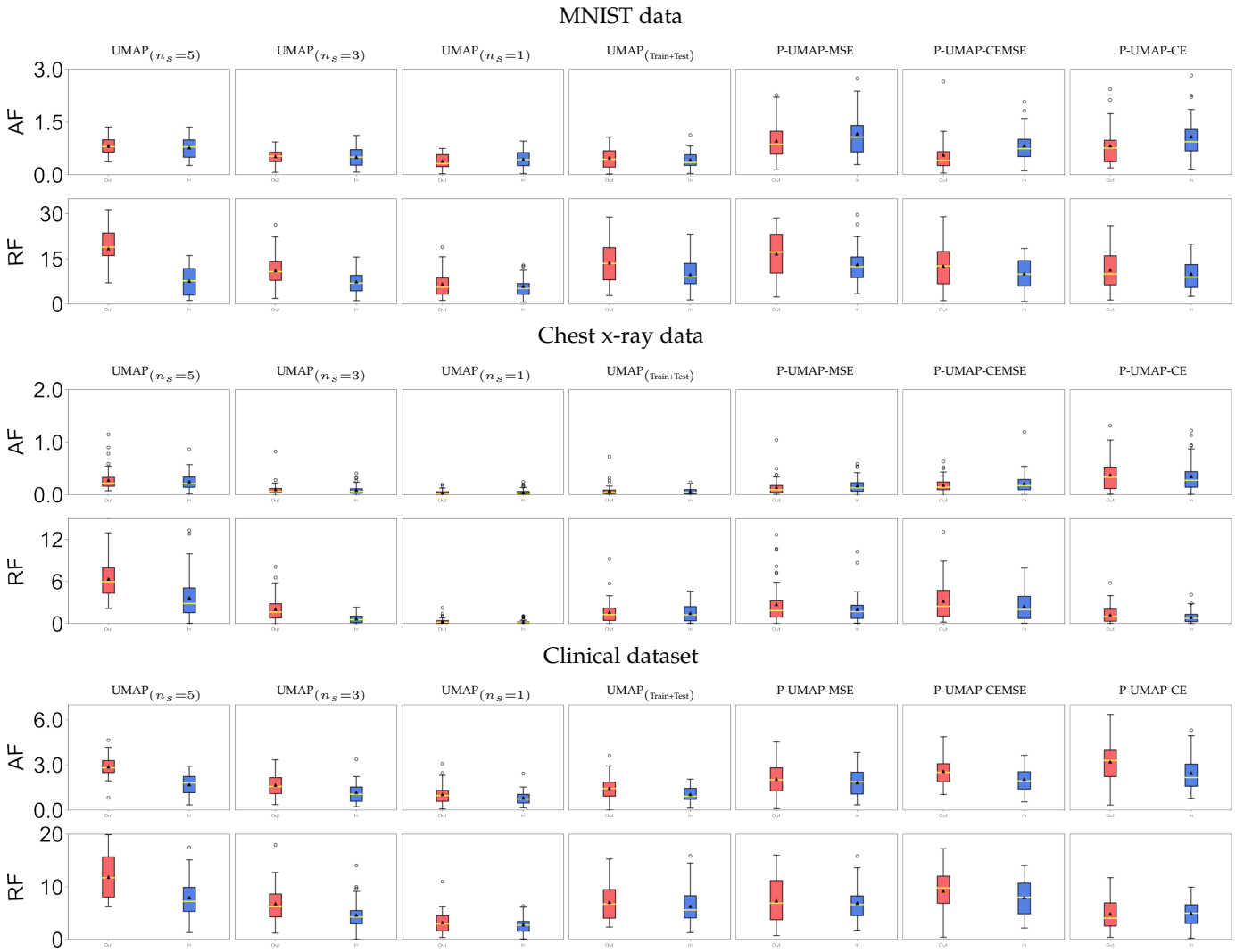


Fig. 8: Disitrbution of attractive forces (AF) and repulsive forces (RF) for MNIST, chest x-ray and clinical datasets. For each dataset, top row shows distribution of attractive forces and bottom row shows distribution of repulsive forces for sets S_1 (outside the periphery, red) and S_2 (inside the periphery, blue), respectively. The boxplots (whiskers) are drawn within the (1.5) interquintile range. The median and mean are indicated by the yellow bar and black tirangle respectively.

the one that produced the highest trustworthiness (more details in Fig. S5 of Supplementary Section S.III). For the parameterized embeddings, we set m_d to 0.1.

There were $24,652$ hospital visits pertaining to shortness of breath, with $15,791$ of those visits resulting in admissions. The corresponding vectors, after converting labels to one-hot encoded data and dropping features with missing entries, are 1003-dimensional. The number of visits in the training split is $19,721$ and in the test split is $4,931$. The two-dimensional embeddings obtained from the bare and parameterized UMAPs are shown in Fig. 6. All the embeddings have the same general structure showing strong separation of patients who were admitted from those who were discharged. The trustworthiness values, given in Table 4, show that P-UMAP-CE provides highest trustworthiness values for all nearest neighbors considered, with P-UMAP-CEMSE the second highest. Interestingly, for training embedding at a very local level, when only 5 nearest neighbors are considered, the original UMAP has a higher trustworthiness score than the versions containing MSE. However, when the training data and then test data are embedded, the second-highest trustworthiness is obtained when n_s is set to 3. Similar to previous datasets, when both training and test data are considered for trustworthiness, the P-UMAP-CE shows superior embedding capabilities. The 2-NN and 5-NN classifiers show similar performance for all the methods, while P-UMAP-CE and P-UMAP-CEMSE are marginally better.

Observations of the repulsion effect are shown in Fig. 7. As before, when both training and test data are considered test points are distributed uniformly within the clusters of baseline UMAP (Fig. 7(a)) but accumulate at the periphery when out-of-sample points are mapped after the training embedding (Fig. 7(b)). This accumulation at the boundary disappears when UMAP employs a lower value of n_s (Fig. 7(c)) and in the parametric implementation P-UMAP-CE (Fig. 7(d)).

TABLE 5: AFR and RFR in terms of $\mathcal{F}_a = |1 - \text{AFR}|$ and $\mathcal{F}_r = |1 - \text{RFR}|$, respectively for different datasets.

	MNIST dataset				Chest x-ray dataset				Clinical dataset			
	$\mathcal{F}_a$	(%I)	$\mathcal{F}_r$	(%I)	$\mathcal{F}_a$	(%I)	$\mathcal{F}_r$	(%I)	$\mathcal{F}_a$	(%I)	$\mathcal{F}_r$	(%I)
UMAP $_{(n_s = 5)}$	0.06	(0.0%)	1.40	(0.0%)	0.07	(0.0%)	0.39	(0.0%)	0.69	(0.0%)	0.49	(0.0%)
UMAP $_{(n_s = 3)}$	0.04	(33.33%)	0.50	(64.29%)	0.08	(-14.29%)	0.78	(-100%)	0.40	(42.03%)	0.47	(4.08%)
UMAP $_{(n_s = 1)}$	0.11	(-83.33%)	0.13	(90.71%)	0.08	(-14.29%)	0.15	(61.53%)	0.29	(57.97%)	0.16	(67.35%)
UMAP $_{\text{(Train+Test)}}$	0.13	(-116.67%)	0.41	(70.71%)	0.01	(85.71%)	0.08	(79.49%)	0.39	(43.47%)	0.12	(75.51%)
P-UMAP-MSE	0.17	(-183.33%)	0.27	(80.71%)	0.11	(-57.14%)	0.13	(66.67%)	0.13	(81.16%)	0.06	(87.76%)
P-UMAP-CEMSE	0.32	(-433.33%)	0.24	(82.86%)	0.07	(0.0%)	0.15	(61.53%)	0.26	(62.31%)	0.16	(67.35%)
P-UMAP-CE	0.24	(-300%)	0.13	(90.71%)	0.05	(28.57%)	0.19	(51.28%)	0.24	(66.22%)	0.02	(95.92%)

4.4 Analyzing Attractive and Repulsive Forces

In this section, we analyze the attractive and repulsive forces of the embedding test points around the periphery of clusters depicted above in more detail and apply the force ratios (AFR and RFR) described in Section 4.2.4. AFR and RFR are applied to two sets of points from the dataset. Likewise, from each dataset, we defined two regions near each other in UMAP $_{(n_s = 5)}$ embeddings: one near/outside the periphery of the clusters and the other inside the periphery of the clusters. The set S_1 is drawn from the test points of the former region, and S_2 is drawn from the test points of the latter region. We ensured that $|S_1| = |S_2|$ for each dataset. The locations where S_1 and S_2 are sampled from are shown in Fig. S8 of the supplementary. It should be noted that the absolute value of the forces is not important here, as by scaling the embedding one can change the absolute value of forces without changing the embedding's characteristics in terms of trustworthiness or k-NN classifier.

The distribution of attractive forces and repulsive forces are shown in Fig. 8. For each dataset, the top row depicts the distribution of attractive forces and the bottom row depicts the distribution of repulsive forces. The boxplots are drawn within the interquartile range (range from first to third quartile), and the whiskers are drawn within the 1.5 interquartile range value. The yellow bar and black triangle indicate the median and mean, respectively. Overall, we can observe that the attractive forces are dispersed equally in both sets for all the methods. However, it is seen that the distributions of repulsive forces have a significant difference between S_1 and S_2 for UMAP $_{(n_s = 5)}$, especially for the MNIST, and Clinical datasets. The distribution of repulsive forces is much broader (the mean is higher and the medians are further from the mean) for points outside the periphery (S_1) than the points inside the periphery (S_2).

It is to be noted again that the nominal value of AFR and RFR is 1.0 when the forces are balanced for both of the sets (S_1 and S_2) sampled from regions close to each other. For ease of understanding, we tabulated the values in terms of $\mathcal{F}_a = |1 - \text{AFR}|$ and $\mathcal{F}_r = |1 - \text{RFR}|$ in Table 5, where the minimum achievable value is 0, thus indicating a 'lower is better' metric. We also tabulate the percent change (%I) of these values considering the baseline of UMAP $_{(n_s = 5)}$ with the positive %I indicating improvement. The objective is to show how disparate UMAP $_{(n_s = 5)}$ is compared to other methods. We considered 30 nearest neighbors to compute $\mathcal{F}_a$ and $\mathcal{F}_r$.

$\mathcal{F}_r$ values have improved over UMAP $_{(n_s = 5)}$ for all the embeddings except for x-ray data for UMAP $_{(n_s = 3)}$. Overall, repulsive forces have become more balanced compared to UMAP $_{(n_s = 5)}$ across all the datasets for the parameterized algorithms. Numerically, for P-UMAP-CE, $\mathcal{F}_s$ is 28.57% and $\mathcal{F}_r$ is 51.28% improved in chest x-ray data over the standard UMAP $_{(n_s = 5)}$. For clinical data the improvement is 66.22% in $\mathcal{F}_a$ and 95.92% in $\mathcal{F}_r$ over UMAP $_{(n_s = 5)}$. For MNIST data, $\mathcal{F}_r$ has improved by 90.71%, however, the improvement is not seen for $\mathcal{F}_a$. This analysis also shows the repulsion effect is greatly reduced in parameterized UMAPs, supporting the results shown by visualization, trustworthiness metric and accumulation in Section 4.3.

5 Conclusion

The original UMAP algorithm is not an online algorithm, in the sense that it cannot accommodate new data points without re-running the embedding from scratch. When it tries to embed out-of-sample test points, they are placed on the periphery of existing clusters. We have demonstrated that this behavior arises from dominant repulsive forces during optimization, regardless of the number of nearest neighbors considered. Reducing this "repulsion effect" (e.g. by making the negative sampling parameter n_s lower than that of training embedding) leads to better embeddings, but the best performance is obtained by parameterizing the mapping. We parameterized UMAP using cross-entropy (CE) loss and mean-squared error (MSE), individually and in combination, and demonstrated its behavior on MNIST digits, chest x-ray images, and clinical data. P-UMAP-CE consistently outperformed the other algorithms, in both visualization and trustworthiness, with the benefits of parameterization increasing as the data became more complex. Analyzing repulsive forces, we further showed that the points that are pushed outside the periphery have disproportionately higher repulsive force than the points that are just inside. We characterized this phenomenon using repulsive force ratio (RFR). Similarly, we also characterized attractive forces of such points using attractive force ratio (AFR). We showed that parameterized UMAPs provide better ARF and RFR values for out-of-sample points compared to that of the original UMAP. The force decomposition used here is a general analysis tool for any dimensionality reduction algorithm, while the parameterized algorithms that result from it should find application in any field that must be updated continuously, e.g. those involving clinical and biomedical data.

Data and Code Availability

The data used in this research are publicly available. The codes used for analysis are uploaded to github: https://

github.com/tariqul-islam/OOS_UMAP/.

ACKNOWLEDGMENTS

The authors would like to thank Hong et el. [15] for making the clinical data and corresponding documentation for parsing the dataset publicly available, which accelerated setting up the experiments with clinical dataset.

REFERENCES

[1] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[2] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[3] G. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *NIPS*, vol. 15. Citeseer, 2002, pp. 833–840.

[4] E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck *et al.*, "Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets," *Cell*, vol. 161, no. 5, pp. 1202–1214, 2015.

[5] D. Kobak and P. Berens, "The art of using t-sne for single-cell transcriptomics," *Nature communications*, vol. 10, no. 1, pp. 1–14, 2019.

[6] J. Cao, M. Spielmann, X. Qiu, X. Huang, D. M. Ibrahim, A. J. Hill, F. Zhang, S. Mundlos, L. Christiansen, F. J. Steemers *et al.*, "The single-cell transcriptional landscape of mammalian organogenesis," *Nature*, vol. 566, no. 7745, pp. 496–502, 2019.

[7] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using umap," *Nature biotechnology*, vol. 37, no. 1, pp. 38–44, 2019.

[8] J. S. Packer, Q. Zhu, C. Huynh, P. Sivaramakrishnan, E. Preston, H. Dueck, D. Stefanik, K. Tan, C. Trapnell, J. Kim *et al.*, "A lineage-resolved molecular atlas of c. elegans embryogenesis at single-cell resolution," *Science*, vol. 365, no. 6459, p. eaax1971, 2019.

[9] L. Chen, X. Li, M. Chen, Y. Feng, and C. Xiong, "The ace2 expression in human heart indicates new potential mechanism of heart injury among patients infected with sars-cov-2," *Cardiovascular research*, vol. 116, no. 6, pp. 1097–1100, 2020.

[10] X. Zou, K. Chen, J. Zou, P. Han, J. Hao, and Z. Han, "Single-cell rna-seq data analysis on the receptor ace2 expression reveals the potential risk of different human organs vulnerable to 2019-ncov infection," *Frontiers of medicine*, pp. 1–8, 2020.

[11] S. Lukassen, R. L. Chua, T. Trefzer, N. C. Kahn, M. A. Schneider, T. Muley, H. Winter, M. Meister, C. Veith, A. W. Boots *et al.*, "Sars-cov-2 receptor ace 2 and tmprss 2 are primarily expressed in bronchial transient secretory cells," *The EMBO journal*, vol. 39, no. 10, p. e105114, 2020.

[12] C. K. Anjinappa and I. Güvenç, "Coverage hole detection for mmwave networks: An unsupervised learning approach," *IEEE Communications Letters*, 2021.

[13] Z. Xu, B. Huang, B. Jia, W. Li, and H. Lu, "A boundary aware wifi localization scheme based on umap and knn," *IEEE Communications Letters*, 2022.

[14] G. J. Berman, D. M. Choi, W. Bialek, and J. W. Shaevitz, "Mapping the stereotyped behaviour of freely moving fruit flies," *Journal of The Royal Society Interface*, vol. 11, no. 99, p. 20140672, 2014.

[15] W. S. Hong, A. D. Haimovich, and R. A. Taylor, "Predicting hospital admission at emergency department triage using machine learning," *PloS one*, vol. 13, no. 7, p. e0201016, 2018.

[16] J. Fleischer and M. T. Islam, "Late breaking abstract-identifying and phenotyping covid-19 patients using machine learning on chest x-rays," *European Respiratory Journal*, 2020.

[17] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on computers*, vol. 100, no. 5, pp. 401–409, 1969.

[18] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[19] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[20] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[21] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th international conference on world wide web*, 2016, pp. 287–297.

[22] L. van der Maaten, "Accelerating t-sne using tree-based algorithms," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[23] Z. Yang, J. Peltonen, and S. Kaski, "Scalable optimization of neighbor embedding for visualization," in *International Conference on Machine Learning*. PMLR, 2013, pp. 127–135.

[24] J. Barnes and P. Hut, "A hierarchical o (n log n) force-calculation algorithm," *nature*, vol. 324, no. 6096, pp. 446–449, 1986.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[26] G. C. Linderman, M. Rachh, J. G. Hoskins, S. Steinerberger, and Y. Kluger, "Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data," *Nature methods*, vol. 16, no. 3, pp. 243–245, 2019.

[27] S. Damrich and F. A. Hamprecht, "On umap's true loss function," in *Proceedings of Advances in Neural Information Processing Systems*, 2021.

[28] J. N. Böhm, P. Berens, and D. Kobak, "Attraction-repulsion spectrum in neighbor embeddings," *Journal of Machine Learning Research*, vol. 23, no. 95, pp. 1–32, 2022.

[29] S. Damrich, J. N. Böhm, F. A. Hamprecht, and D. Kobak, "From *t*-sne to umap with contrastive learning," in *International Conference on Learning Representations*, 2023.

[30] H.-K. Ko, J. Jo, and J. Seo, "Progressive uniform manifold approximation and projection." in *EuroVis (Short Papers)*, 2020, pp. 133–137.

[31] D. A. Senanayake, W. Wang, S. H. Naik, and S. Halgamuge, "Self-organizing nebulous growths for robust and incremental data visualization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4588–4602, 2020.

[32] M. T. Islam and J. W. Fleischer, "Manifold-aligned neighbor embedding," in *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[33] L. Van Der Maaten, "Learning a parametric embedding by preserving local structure," in *Artificial Intelligence and Statistics*, 2009, pp. 384–391.

[34] K. Bunte, M. Biehl, and B. Hammer, "A general framework for dimensionality-reducing data visualization mapping," *Neural Computation*, vol. 24, no. 3, pp. 771–804, 2012.

[35] A. F. Duque, S. Morin, G. Wolf, and K. Moon, "Extendable and invertible manifold learning with geometry regularized autoencoders," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 5027–5036.

[36] T. Sainburg, L. McInnes, and T. Q. Gentner, "Parametric umap embeddings for representation and semisupervised learning," *Neural Computation*, vol. 33, no. 11, pp. 2881–2907, 2021.

[37] Z. Zhou, X. Zu, Y. Wang, B. P. Lelieveldt, and Q. Tao, "Deep recursive embedding for high-dimensional data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 2, pp. 1237–1248, 2021.

[38] P. G. Poličar, M. Stražar, and B. Zupan, "Embedding to reference t-sne space addresses batch effects in single-cell classification," in *International Conference on Discovery Science*. Springer, 2019, pp. 246–260.

[39] L. McInnes, J. Healy, N. Saul, and L. Grossberger, "Umap: Uniform manifold approximation and projection," *The Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[42] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[43] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.
[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[45] J. Venna and S. Kaski, "Neighborhood preservation in nonlinear projection methods: An experimental study," in *International Conference on Artificial Neural Networks*. Springer, 2001, pp. 485–491.
[46] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.
[47] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.
[48] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.
[49] RSNA, *RSNA pneumonia detection challenge*, 2018. [Online]. Available: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge
[50] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
[52] N. C. Hurley, A. D. Haimovich, R. A. Taylor, and B. J. Mortazavi, "Visualization of emergency department clinical data for interpretable patient phenotyping," in *2019 4th IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies*, 2019.