Hierarchical Nearest Neighbor Graph Embedding for Efficient Dimensionality Reduction

M. Saquib Sarfraz [*1,2], Marios Koulakis [*1], Constantin Seibold[1], Rainer Stiefelhagen[1]
[1] Karlsruhe Institute of Technology, [2] Mercedes-Benz Tech Innovation
[*] Equal contribution

Abstract

Dimensionality reduction is crucial both for visualization and preprocessing high dimensional data for machine learning. We introduce a novel method based on a hierarchy built on 1-nearest neighbor graphs in the original space which is used to preserve the grouping properties of the data distribution on multiple levels. The core of the proposal is an optimization-free projection that is competitive with the latest versions of t-SNE and UMAP in performance and visualization quality while being an order of magnitude faster at run-time. Furthermore, its interpretable mechanics, the ability to project new data, and the natural separation of data clusters in visualizations make it a general purpose unsupervised dimension reduction technique. In the paper, we argue about the soundness of the proposed method and evaluate it on a diverse collection of datasets with sizes varying from 1K to 11M samples and dimensions from 28 to 16K. We perform comparisons with other state-of-the-art methods on multiple metrics and target dimensions highlighting its efficiency and performance. Code is available at https://github.com/koulakis/h-nne

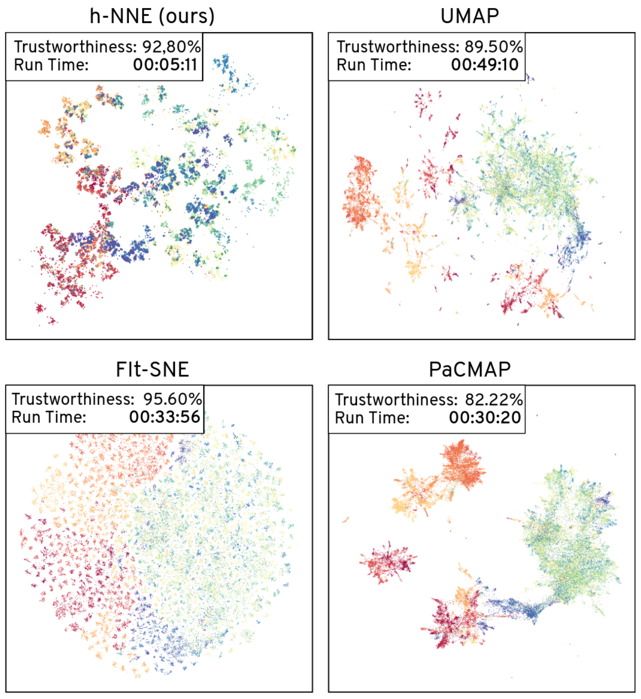


Figure 1. Visualization of the entire ImageNet dataset. Speed and embedding quality of dimension reduction methods.

1. Introduction

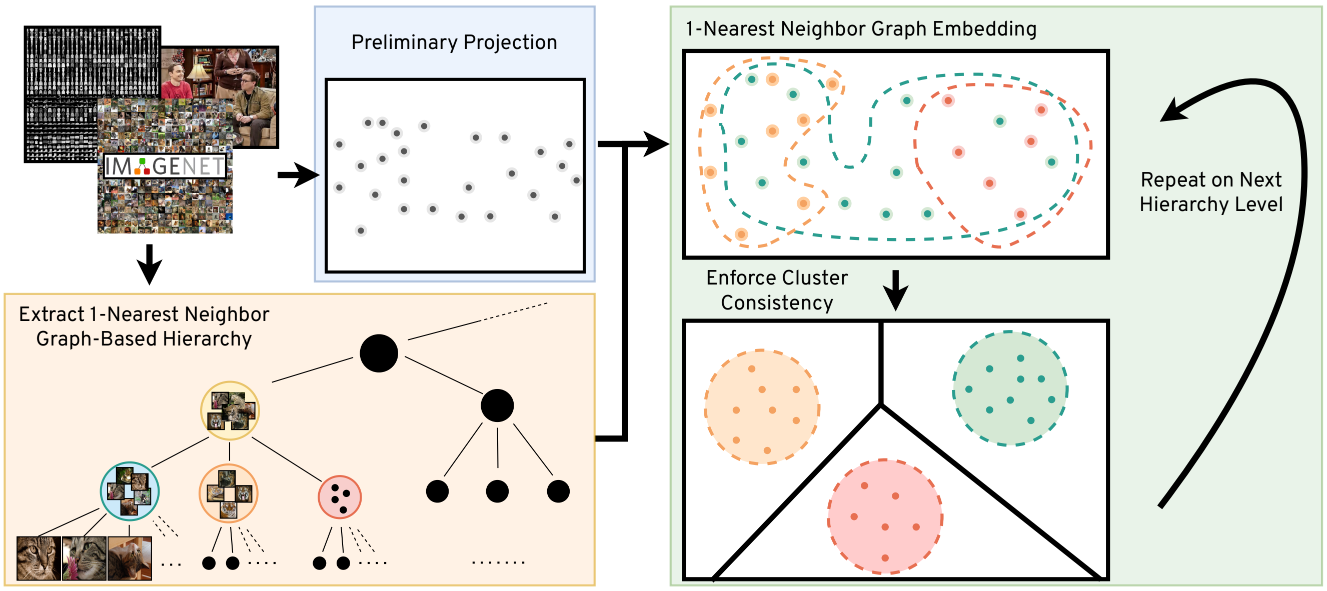
Dimensionality reduction techniques are now increasingly used in many fields of science and have to cope with an ever increasing size of real-world datasets. It plays an important role both for visualization and processing of high dimensional data. Much of current research is focused on finding unsupervised algorithms that are both scalable to massive data and are able to preserve the structure of data in less dimensions. Most of them attempt to retain the local or global structure of the data by optimizing over pairwise distances in the target space. Two main directions for the current dimension reduction techniques can be identified with respect to how such local or global neighborhood is preserved in terms of the distances. Methods such as PCA [15], MDS [18] and Sammom mapping [31] try to preserve the global distances among all samples in the data. Whereas more recent popular methods such as t-SNE [35,36], LargeVis [33], and UMAP [26,30] seek to additionally preserve the local structure e.g. by preserving the distance relations in the k-neighborhood of each data sample. To retain such relations, these methods generally have to solve an optimization problem with the goal of matching the distribution of distances in the target space with their distribution in the original space. For instance, t-SNE minimizes the Kullback-Leibler divergence between distributions of k-nearest neighbor (k-NN) distances fitted in the high and low-dimensional space. Similarly, the more recent method UMAP optimizes the embedding in the target space with the goal of preserving the 1-skeleton of fuzzy simplicial sets constructed in the original space. Such optimizations are computationally expensive in nature and account for the main complexity of these algorithms, thus limiting their run-time performance on large scale datasets.

In this paper, we present a different approach which instead of relying on point-level optimization, captures multi-stage NN properties of the data and, using those, projects points in a simple algorithmic way. The main tools used to build this structure are Nearest Neighbor Graphs (NNGs) which have been well studied. In [12] Epstein *et al*. show that for a 1-NNG, its NN relations are well preserved in a low dimensional space when the edges are placed on a monotone logical grid [8]. An effective strategy for embedding the NNG into an l-dimensional grid is to embed the individual components of the graph separately. The connected components of NNGs capture clusters of samples. Recursively building 1-NNGs on the previously obtained connected components provides a hierarchical view on how samples are merged together at successive levels. Considering each connected component as a node in the hierarchy one can identify complete paths on how these nodes and their associated samples successively merge together from bottom to top. Such a hierarchical node graph provides a view of data in terms of how the local neighborhood is distributed in the high dimensional space. After an inexpensive preliminary projection of the high dimensional data on a desired low dimensional space we can use the original hierarchical node graph in the target space to enforce the local structure directly. We achieve this with a fast recursive top-down approach by moving the clusters of samples towards these nodes starting at the top level with the least number of nodes and moving progressively downward to reach the bottom *i.e*. the finer level.

Since our proposal is rooted in obtaining the Hierarchical 1-Nearest Neighbor graph based Embedding, we term the method **h-NNE**. Figure 1 depicts an example of embedding the ResNet-50 features of the full ImageNet dataset. As seen, in comparison to the current state-of-the-art methods, not only do we achieve competitive embedding quality - indicated by the trustworthiness metric - but also a significantly faster run-time. To summarize, our main contribution is an alternative dimensionality reduction and visualization method which does not rely on expensive optimization methods. This makes it operate at a magnitude faster than existing methods, without requiring hyperparameter tuning, and maintaining similar performance. In the following sections, before delving into the proposed method, we will discuss the related works to place it in context and followed by experiments and comparisons with the state-of-the-art on a diverse collection of datasets.

2. Related work

Most of the current state-of-the-art unsupervised dimension reduction techniques aim at preserving the local pairwise distances in a projected manifold. Hinton and Roweis [14] introduced the Stochastic Neighborhood Embedding (SNE), which creates a low-dimensional embedding by enforcing the conditional probabilities (euclidean distances converted to similarities) in the lower dimension to be similar to those in the higher dimension. This is achieved by fitting Gaussian distributions on the samples and matching them to distributions in the lower dimension. Building on SNE, t-Distributed Stochastic Neighbor Embedding (t-SNE) [36] substituted the Gaussian distribution, used in the low-dimensional space, with long-tailed t-distributions. t-SNE is inherently computationally expensive. Follow-up works of t-SNE such as Barnes-Hut t-SNE [35], viSNE [2], FIt-SNE [20, 21], and opt-SNE [6], have further improved t-SNE so it can converge faster and scale better to larger datasets. Much of the followup work follows the same direction. For example, LargeVis [33] and the current state-of-the art methods like UMAP [26] and PaCMAP [38] have objective functions building on that of t-SNE. These methods focus on improving efficiency and preservation of more global structure along with the local structure. This is commonly achieved by construction and operations on weighted graphs. Typically a k-NN graph is constructed in the original space and then used to enforce the k-neighborhood of each sample point in a preliminary projected space. All current state-of-the-art methods such as t-SNE [36], LargeVis [33], UMAP [26] and PaCMAP [38] are heavily related in terms of this underlying process. A more thorough discussion on this k-NN graph optimization view is provided in McInnes *et al*. [26] that compares t-SNE, LargeVIs and UMAP optimization equations in this context. The difference between these techniques comes from the changes in the objective function that is used for optimizing the projected embedding. For instance a common goal of optimization is to attract together samples that lie in the same k-neighborhood and repulse them from far away samples. If viewed in terms of the k-NN graph layout this amounts to defining and using a set of attractive forces applied along edges and a set of repulsive forces applied among vertices. This is a non-convex optimization problem and convergence is achieved by carefully changing the attraction and repulsion forces through gradient descent based learning. In effect it is contrastive learning by using samples in k-neighborhoods as positives to the current data point and sampling negatives from the rest. To reduce cost, UMAP employs negative sampling to pick a subset of samples for the repulsive force whereas the more recent PaCMAP defines highly-weighted near-pairs and mid-near pairs to help preserve more structure. A related strategy is used by triplet constraint methods such as TriMap [1] which is initialized with the low dimensional PCA embedding. This embedding is then modified using a set of carefully selected triplets from the high-dimensional representation. Finally, t-SNE and UMAP methods have



Preliminary Projection
1-Nearest Neighbor Graph Embedding
Repeat on Next Hierarchy Level
Enforce Cluster Consistency
Extract 1-Nearest Neighbor Graph-Based Hierarchy
IMAGENET

Figure 2. A summary of the h-NNE projection method.

their parametric versions [30, 34] where they train an MLP with a similar objective function. The parametric methods simplify the projection of new datapoints. Another important consideration in these methods is their reliance on user supplied hyperparameters. Apart from specifying the number of neighbors k to construct the graph, several other optimization specific parameters are also required.

Our approach is a major shift in contrast to these current methods. Instead of building a weighted k-NN graph, we create a clustering hierarchy by recursively building 1-NN graphs with static edge links. This hierarchy is then used to move samples in the lower dimensional space without requiring the use of gradient descent based optimization. Simultaneously, this removes the reliance on specific hyperparameters. This provides us with a highly efficient and scalable dimension reduction method.

3. The h-NNE algorithm

Our projection algorithm consists of three main steps: building a tree hierarchy based on 1-NNGs, computing a preliminary projection with an approximate version of PCA and adjusting the projected point locations based on the constructed tree. The projected point location adjustment can be enhanced with an optional inflation step which can be used to improve visualization. In the following sections, we will elaborate on each step and provide some evidence of their validity. Figure 2 gives an overview of the method.

3.1. Nearest neighbor graph based hierarchy

Several projection methods start by defining a structure over the data which encodes the relative positions of points and then project in a way that preserves this structure. For example, UMAP relies on a weighted graph encoding nearest neighbor relations while t-SNE uses a collection of local distributions based again on the nearest neighbor relations. In our case, we strive for a structure which captures both local neighbor properties of points and global clustering properties. In order to achieve this with a low computational cost, and at the same time keep the approach simple and parameter free, we build a hierarchy based on 1-NN relations between points. This approach is inspired by classical work on nearest neighbor graphs such as [12] and the FINCH clustering method [32].

Assume that our dataset is $\mathbf{X} = \{\mathbf{x}_i\}_{i \leq N}$, where $\mathbf{x}_i \in \mathbb{R}^D$. The first step in constructing the hierarchy entails building $NNG(\mathbf{X})$ which is a directed graph that connects each point to its nearest neighbor. This can be performed by using any nearest neighbor or approximate nearest neighbor algorithm. Next, we identify the connected components of $NNG(\mathbf{X})$, denoted by $\{NNG_i(\mathbf{X})\}_{i \leq g_0}$, which form directed graphs with all edges pointing to a single bi-root. For each graph $NNG_i(\mathbf{X})$, we compute its centroid $\mathbf{c}_i^{(0)} = \frac{1}{g_0} \sum_{\mathbf{x} \in NNG_i(\mathbf{X})} \mathbf{x}$ and thus form a new set of points $\mathbf{C}^{(0)} = \{\mathbf{c}_i^{(0)}\}_{i \leq g_0}$. We then repeat the same process of computing $NNG(\mathbf{C}^{(0)})$ and its components' centroids to derive $\mathbf{C}^{(1)}$ and continue until we reach the smallest set of centroids $\mathbf{C}^{(l)}$ which contains at least three points. The NNG hierarchy is then the tree $T_{NNG}(\mathbf{X}) = \langle \bigcup_{i \leq l} \mathbf{C}^{(i)} \cup \mathbf{X}, E \rangle$, where each centroid is connected to each of the points of the NNG component which corresponds to it. Figure 3 displays a single step of this iterative process.

In comparison with k-NNGs for $k > 1$, 1-NNGs are quite small in size, which make them well-suited to construct a fine-grained hierarchy with several levels. A simple rule of thumb is that the number of points decreases on average by a factor of 0.31 on every step. This can be extracted from the following theorem assuming that, at least locally, the data and centroids on higher levels are uniformly distributed. Though the validity of the assumption is hard to verify, we notice that this rule holds for all observed datasets.

Theorem 1 (Eppstein, Paterson, Yao [12])**.** *The expected number of components in $NNG(\mathbf{X})$ for a uniform random point set $\mathbf{X}$ in a unit square is asymptotic to approximately* $0.31|\mathbf{X}|$.

3.2. Preliminary linear embedding

Though the clustering tree structure is enough to produce a projection respecting the partitions of the original dataset on separate levels, the relative positions of the points contained in the graph components $NNG_i(\mathbf{X})$ and $NNG_i(\mathbf{C}^{(k)})$ need to be determined in the target space. One could use a random projection, or even start with random points, but there is an extra gain in both preservation scores and in visual quality when using some meaningful initial projection. We choose to use PCA and to accelerate its computation, we estimate the covariance matrix of the data using the centroids $\mathbf{C}^{(i)}$ of a predefined level of the hierarchy. One could as well sub-sample the original point set $\mathbf{X}$, but we notice that using the centroids increases stability and avoids deviation of the initialization between runs. We experimentally verify that this approximation of principal components produces results comparable to using PCA on the full data. We provide this analysis in the supplementary.

We choose the level of centroids to be the lowest level of the hierarchy such that all levels above have cardinality less than 1000. This implies that if the dataset is small, thus for all i, $|\mathbf{C}^{(i)}| < 1000$, then the PCA will directly be computed on $\mathbf{X}$. The advantage of this approximation is that we can reduce the computational cost of PCA from $\mathcal{O}(N \cdot D^2)$ to $\mathcal{O}(D^2)$ with N replaced with a factor of 1000.

Once the eigenvectors of PCA are computed, say in a matrix V, then all points of $\mathbf{X}$ and all centroids $\mathbf{C}_i$ are projected from the higher dimension D to the lower dimension d by multiplying with this single shared matrix. We denote the projections of such points by a tilde superscript, for example $\tilde{\mathbf{c}}_i^{(k)}$ for centroids and $\tilde{\mathbf{x}}_i$ for points of the dataset.

3.3. Hierarchical point translation

This is the central part of the algorithm and the goal is to move the points hierarchically so that they occupy the projection space $\mathbb{R}^d$ following the tree $T_{NNG}(\mathbf{X})$ in a way that the 1-NN relationships are preserved over all levels. Once we have a preliminary projection for all points and centroids, we start from $\mathbf{C}^{(l)}$ and consider its projected centroids $\{\tilde{\mathbf{c}}_i^{(l)}\}_{i \leq g_l}$. Those centroids form a Voronoi tessellation of $\mathbb{R}^d$ and an ideal way to place the lower level projected centroids $\{\tilde{\mathbf{c}}_i^{(l-1)}\}_{i \leq g_{l-1}}$ would be to select for each $\tilde{\mathbf{c}}_i^{(l)}$ the centroids of level $l-1$ which correspond to it, translate them so that they are centered around $\tilde{\mathbf{c}}_i^{(l)}$ and finally spread them to occupy the corresponding Voronoi cell. In order to perform this process in an efficient and easy to vectorize way, we choose to use the already known distance $d_i^{(l)}$ of each $\tilde{\mathbf{c}}_i^{(l)}$ to its nearest neighbor in $\tilde{\mathbf{C}}^{(l)}$ and then scale the translated, lower level centroids to a d-ball of radius $\frac{1}{3}d_i^{(l)}$. This distance guarantees that points belonging to neighboring centroids will not form nearest neighbor relationships cross-centroids, thus preserving the separation encoded in $T_{NNG}(\mathbf{X})$. Figure 4 illustrates this process.

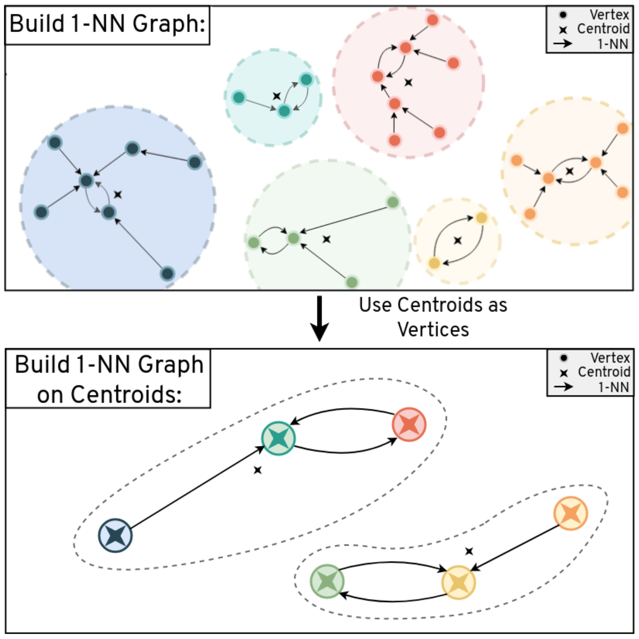


Figure 3. The induction step in building $T_{NNG}(\mathbf{X})$. All 1-NNG components are mapped to their centroids which form the basis for the next step.

Once the points of $\tilde{\mathbf{C}}^{(l-1)}$ are placed around the points of $\tilde{\mathbf{C}}^{(l)}$, we use them to translate the points of $\tilde{\mathbf{C}}^{(l-2)}$ around them, the same way as before. This step is repeated until we reach the level of $\mathbf{X}$ which forms the final projection.

There is still one issue we need to address. This is the fact that though the radius $\frac{1}{3}d_i^{(l)}$ guarantees the separation of neighboring centroids on one step, it could be the case that the borders of this d-ball are crossed by points moved on a later step of the iteration (see again figure 4). Below we compute a shrinking coefficient for this radius, such that this guarantee still holds for the points moved in later steps.

Lemma 1. *Given a d-ball $B(\mathbf{c}_i^{(k)}, r)$ centered on a centroid, all points belonging to $\mathbf{c}_i^{(k)}$ translated with the h-*

NNE algorithm with radii multiplied a factor of $\frac{3}{5}$, lie inside $B(\mathbf{c}_i^{(k)}, r)$.

Proof. Assume that a factor s is used to reduce the computed radii on each step of h-NNE. On the first step, all lower level centroids are translated and scaled so that they lie in a d-ball of radius sr. The worst case scenario for the next step is that there a two antidiametrical points $\mathbf{c}_1^{(k)}$, $\mathbf{c}_2^{(k)}$ which are nearest neighbors. In that case, the points of the next step belonging to $\mathbf{c}_1^{(k)}$ will be placed inside a d-ball around it of radius $\frac{2}{3}sr$ since $d(\mathbf{c}_1^{(k)}, \mathbf{c}_2^{(k)}) = 2sr$. This means that the largest distance of any of those points to $\mathbf{c}_i^{(k)}$ is $sr + \frac{2}{3}s^2r$. By recursively computing those worst case scenarios, we get that for infinite steps of the algorithm, the most distant point will be placed in a distance of at most

$$\sum_{j\in\mathbf{N}}(\frac{2}{3})^j s^{j+1} r = sr\sum_{j\in\mathbf{N}}(\frac{2}{3}s)^j = \frac{sr}{1-\frac{2}{3}s} \qquad (1)$$

Therefore, in order for all points to lie inside the original ball $B(\mathbf{c}_i^{(k)}, r)$, we need that $\frac{sr}{1-\frac{2}{3}s} \leq r$ from which we get that $s \leq \frac{3}{5}$. □

The above bound guarantees that if we place new points in a ball of radius $\frac{3}{5} \cdot \frac{1}{3} = 0.2$ times the distance to the nearest neighbor, then the nearest neighbors of points will be restricted in the clusters formed in the NNG hierarchy. In practice the worst case scenario of antidiametrical points does not occur so often. Even in low dimensions where points are more dense one can use radii of $\frac{1}{3}$ of the 1-NN distance or even more without noticeable drop in k-NN preservation. This can be particularly useful for visualization, as it can help make plots more spread on the plane or 3D space.

Point cluster inflation for visualization purposes. The use of a single linear projection for all points can result in cluttered point clusters when they are not well aligned to the global principal components used to project. Though this has minimal impact to performance, it leads to poorer visualization which contains artifacts from this initial projection. In order to enhance the shape of images without sacrificing speed and without adding new hyper-parameters, we add the option to inflate potentially squeezed point clusters using six local rotations with angles equally distanced in the interval $[0, \frac{\pi}{2}]$, followed by a scaling and the inverse rotation. This results in an output almost equivalent to that of rotating the clouds to their PCA principal components, scaling them, and then rotating them back to the original orientation but much less computationally expensive.

Projecting new points. The projection of new points can be performed by repeating the same algorithm as before, just for the individual points and by descending only the second level of the hierarchy. If $\mathbf{x}$ is the new point, we

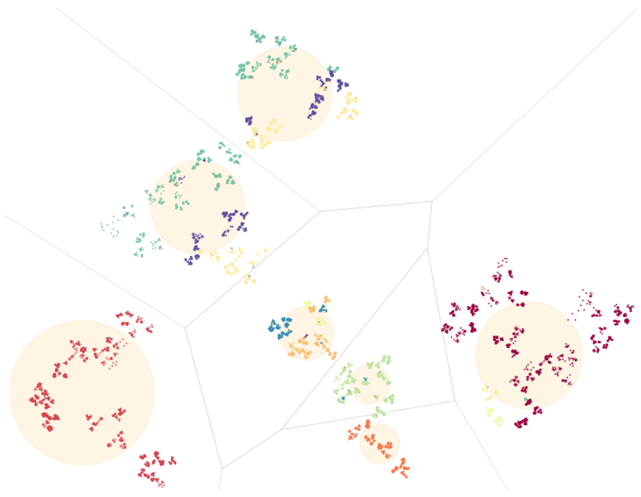


Figure 4. Voronoi cells of the top level centroids in MNIST. The circles have radius $\frac{1}{3}$ the distance from the nearest neighbor and the points are projected with a shrinking factor of exactly this factor of $\frac{1}{3}$. One can notice that the final points cross the boundaries of the circle. Nevertheless, the density of the data in 2D result to a situation where no severe overlaps appear.

first identify the closest centroid in $\mathbf{C}^{(1)}$ using our ANN algorithm of choice. Then the point is transformed by scaling, applying the pre-computed PCA and normalizing the position of the point relative to the centroid based on the corresponding radius. If point cluster inflation was used in the original projection, then the relevant rotations and scalings are also performed before the final normalization step.

3.4. Computational complexity

There are three steps in h-NNE which make up its complexity: the construction of the $T_{NNG}(\mathbf{X})$ tree, the preliminary PCA projection and the hierarchical point translation. Since each component of the NNG graph contains at least two points, the height of $T_{NNG}(\mathbf{X})$ is $\mathcal{O}(\log N)$. That implies that given that the computation of approximate 1-NN is at least $\mathcal{O}(N)$, no matter which method is used, the total complexity is equal to the complexity of a single Approximate Nearest Neighbor (ANN) step including any preparation on the dataset (e.g. building indexing trees) and querying once on all points of the dataset. We denote the ANN complexity with $\mathcal{O}(ANN(N, D))$. The PCA projection step requires $\mathcal{O}(Dd^2)$, given we have fixed the number of used samples to a constant number of points. Finally, the point translation steps are $\mathcal{O}(N \log Nd + ANN(N, d))$, again because of the logarithmic height of the tree, the usage of group by operations, and the fact we need to compute nearest neighbors to find the radius of the d-balls where clusters are expanded. Thus, overall, the algorithm has $\mathcal{O}(ANN(N, D) + Dd^2 + N \log Nd)$ complexity.

The complexity $ANN(N)$ is not easy to compute. In

	Higgs [3]	Google News [27]	COIL 20 [28]	CIFAR-10 [17]	F-MNIST [39]	ImageNet [10]	BBT [5]	Buffy [5]	MNIST [19,23]	
Type	Sensor	Text	Objects				Videos		Digits	
#Classes	2	-	20	10	10	1000	5	6	10	
#Samples	11M	3M	1440	60K	70K	1.2M	200K	206K	70k	8M
Dimension	28	300	16384	3072	784	2048	2048	2048	784	784
Features	Measurements	Word2Vec	Pixels	Pixels	Pixels	ResNet50	ResNet50	ResNet50	Pixels	Pixels

Table 1. Datasets of different size ranging from 1400 to 11 million samples in 28 to 16384 dimensions used in experiments.

the worst case scenario, it could be N^2 when using linear search. Some of the older exact methods [7, 29] achieve $N \log N$ complexity, but unfortunately scale exponentially on D. Approximate methods such as HNSW [24], NNDescent [11] and ScaNN [13] achieve good performance on real-world datasets but support it with experimental evidence and no complexity bounds. In our current implementation we use PyNNDescent [25] which is a tuned version of NNDescent. The empirical complexity of NNDescent is approximately $O(N^{1.14})$ for datasets with small intrinsic dimensionality. Finally, an interesting analysis [4] provides synthetic families of datasets where NNDescent attains a complexity of $O(N^2)$ and $O(N \log N)$ respectively.

4. Experiments

We demonstrate h-NNE on diverse datasets of different size which cover domains such as sensor data, text, digits, videos and objects. We first introduce the datasets and performance metrics, followed by a thorough comparison of the proposed method to current state-of-the-art algorithms.

4.1. Evaluation

Datasets. The datasets are summarized in Table 1. Apart from some commonly used datasets we also include few to test against size and dimensions. Altogether, we use 9 datasets ranging from 1440 to 11 million samples in 28 to 16384 dimensions. We provide more details for each dataset in the supplementary.

Metrics. A well studied issue in dimensionality reduction is how to balance local and global structure preservation [9, 16]. Methods like t-SNE are well known for their local structure preservation. More recent methods such as UMAP [26, 30] and PaCMAP [38] strive to preserve both local and global structure. We therefore evaluate all methods on the considered datasets in these two aspects.

Local structure preservation: commonly measured by leave-one-out cross validation using the *k-NN classifier*. **k-NN accuracy** is a standard evaluation method for dimension reduction methods as this measures if the classification accuracy based on neighborhoods would remain close to that in the original space. As in previous studies, we use a 10-fold stratified cross-validation to measure k-NN accuracy on a varying number of k values. Another closely related local structure preservation metric is Trustworthiness. **Trustworthiness** [37] penalizes for each point every one of its k nearest neighbors in the embedding space by the amount its rank in the original space exceeds k. Trustworthiness is defined as

$$T(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i^k} \max(0, (r(i,j) - k))$$

and is scaled between 0 and 1, with 1 being more trustworthy. We use scikit-learn's implementation with $k = 5$.

Global structure preservation: To measure the preservation of global structure i.e. relative positioning of individual neighborhoods Wang *et al.* [38] proposed a metric that measures how well the distribution of distances between class centers in the original space are preserved in the embedding space. It is obtained by computing the centroids and forming all possible triplets between them. The metric **Triplet Centroid Accuracy** measures the percentage of triplets whose relative distance in the high- and low-dimensional spaces maintain their relative order.

	h-NNE	t-SNE	fIt-SNE	UMAP	PaCMAP
COIL20	0.994	0.998	**0.998**	0.996	0.985
MNIST	0.983	0.989	**0.991**	0.958	0.950
F-MNIST	0.981	0.991	**0.992**	0.977	0.966
CIFAR10	0.907	0.927	**0.932**	0.829	0.818
BBT	0.982	0.99	**0.99**	0.966	0.958
Buffy	0.976	0.988	**0.99**	0.954	0.952
ImageNet	0.928	–	**0.956**	0.895	0.822
HIGGS	0.849	–	**0.979**	0.909	0.899
MNIST 8M	**0.967**	–	0.956	–	0.945

Table 2. Local structure preservation : Trustworthiness.

	h-NNE	t-SNE	fIt-SNE	UMAP	PaCMAP
COIL20	**0.799**	0.778	0.769	0.705	0.758
MNIST	0.671	0.711	0.748	**0.804**	0.713
F-MNIST	**0.925**	0.784	0.848	0.848	0.866
CIFAR10	0.911	0.921	0.929	0.927	**0.932**
BBT	0.644	0.600	**0.711**	0.556	0.666
Buffy	**0.857**	0.571	0.543	0.657	0.704
ImageNet	0.604	–	0.639	0.605	**0.646**
MNIST 8M	**0.774**	–	0.744	–	0.735

Table 3. Global structure preservation: Centroid Triplet Accuracy.

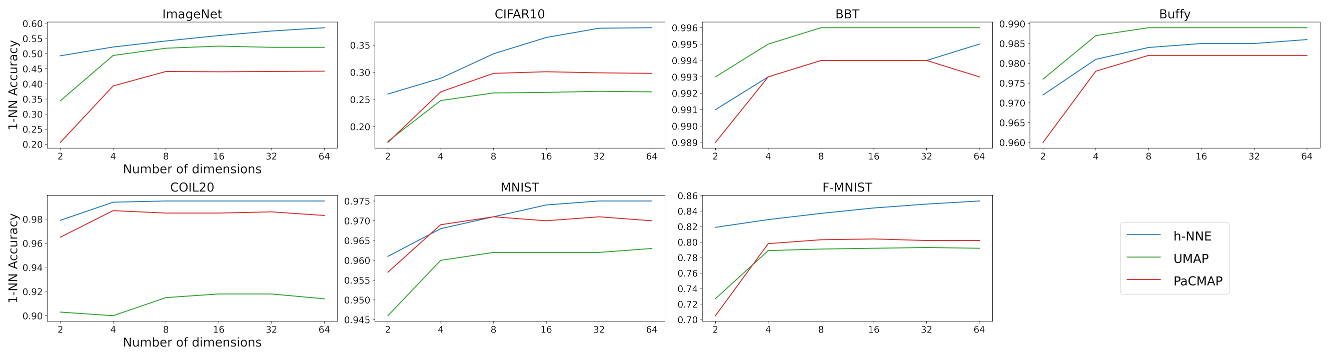


Figure 5. Projection on different dimensions: KNN accuracy for 2, 4, 8, 32 and 64 dimensions.

4.2. Comparison with state-of-the-art

We compare with representative current best methods for unsupervised dimensionality reduction and visualization. t-SNE [36], owing to its remarkable local structure preserving properties and long standing follow up work, is a default choice for 2D visualizations. It has, however, high compute overhead. We use its optimized version Barnes-Hut t-SNE [35] in our comparison. Since this still does not easily scale to data beyond 200K in size, we also include a much faster and scalable variant FFt-accelerated Interpolation-based t-SNE (FIt-SNE) [22]. FIt-SNE is the currently best t-SNE method in terms of speed and performance. Of note, t-SNE is limited in that its running time increases exponentially to the number of dimensions and though FIt-SNE could in principle support higher dimensions it's official implementations do not support it. Among other recent methods, we include the popular UMAP [26] and a very recent method, PaCMAP [38], which builds upon the strengths of t-SNE and UMAP.

Table 2 provides the trustworthiness scores of all methods on the datasets. t-SNE and FIt-SNE both are quite good with respect to preserving the local structure. h-NNE follows closely to t-SNE methods in terms of the local structure preservation. UMAP and PaCMAP perform on par though slightly worse than h-NNE. We provide the k-NN classifier accuracy and comparisons in the supplementary.

On the global structure preservation metric as shown in Table 3 the centroid triplet accuracy for h-NNE embeddings is on par as well across all datasets. Since these metrics require ground truth labels of the datasets the results on Google News dataset cannot be computed. Similarly, the centroid triplet metric on HIGGS is not evaluated as it has only two classes. These results indicate that, overall, h-NNE is highly competitive with these current methods in terms of both local and global structure preservation.

Projections of varying dimensions. There is no restriction on the target dimension for h-NNE. Figure 5 demonstrates projection on different dimensions. For this experiment we include 7 datasets and compare with UMAP and PaCMAP as only these support higher dimensions. Here k-NN ($k = 1$) accuracy across different datasets shows that h-NNE has a consistent upward trend in performance when moving to higher dimensions.

Computational performance comparisons. Benchmarks on all datasets were performed on a workstation with an AMD Ryzen Threadripper 2990X 32-core processor with 128 GB RAM. Table 4 provides the total run-time of each algorithm in comparison. As expected t-SNE was not able to scale on large data size and UMAP ran out of memory on 8 million 784 dimensional data. PaCMAP and FIt-SNE - the fast alternative approach to t-SNE which is a highly optimized and parallelized C++ implementation - both ran on all datasets. h-NNE has a clear advantage in terms of speed and it scales really well on the size of the dataset, reaching speedups well above ten-fold on the larger datasets. The time efficiency of our method comes from the approach itself (i.e. not relying on expensive gradient decent-based embedding optimizations) and could be sped-up further with similar parallelized implementations. On the HIGGS data with 11M samples h-NNE takes ~ 13 minutes vs almost 3 hours for FIt-SNE and $4 - 5$ hours for UMAP and PaCMAP. Since all these existing methods need to build weighted k-NN graphs with the requirement to store and access pairwise distance floats for their optimizations, their memory requirement is very high. For instance as opposed to h-NNE none of these methods could run on a machine with 64 GB ram at million scale. For more details on the computational complexity we refer to section 3.4.

Exploring unlabeled data. We see an advantage of h-NNE in exploring unlabeled data because of its hierarchical nature. On the one hand, different clusters of the data are visually separated, and on the other hand, we have cluster-labels generated on all levels of the hierarchy. In figure 6 we color the Google News dataset projection with those cluster-labels on two different levels and demonstrate a zooming process where we isolate a cluster of similar word vectors.

Limitations. One limitation of our algorithm is its dependence on the hierarchical NNG structure we build. This

Dataset	#S	Dim	h-NNE (ours)	t-SNE	UMAP	PaCMAP	FIt-SNE
COIL20	1440	16384	**00:01**	00:08	00:18	00:02	00:07
MNIST	70K	784	**00:09**	05:28	00:54	00:37	01:24
F-MNIST	70K	784	**00:07**	05:37	00:58	00:38	01:20
CIFAR10	60K	3072	**00:14**	08:35	01:02	00:39	02:24
BBT	200K	2048	**00:58**	57:32	04:08	01:38	04:50
Buffy	206K	2048	**00:50**	01:01:00	04:18	01:42	05:07
Google News	3M	300	**03:34**	–	01:19:05	01:14:04	52:16
ImageNet	1.2M	2048	**05:11**	–	49:10	30:20	39:29
HIGGS	11M	28	**12:25**	–	03:53:04	05:07:34	02:49:28
MNIST_8M	8M	784	**21:52**	–	–	03:08:47	02:56:14
Framework			Python	Python	Python	Python	C++

Table 4. Run-time comparison of h-NNE: We report the run time in HH:MM:SS and MM:SS. – denotes out of memory.

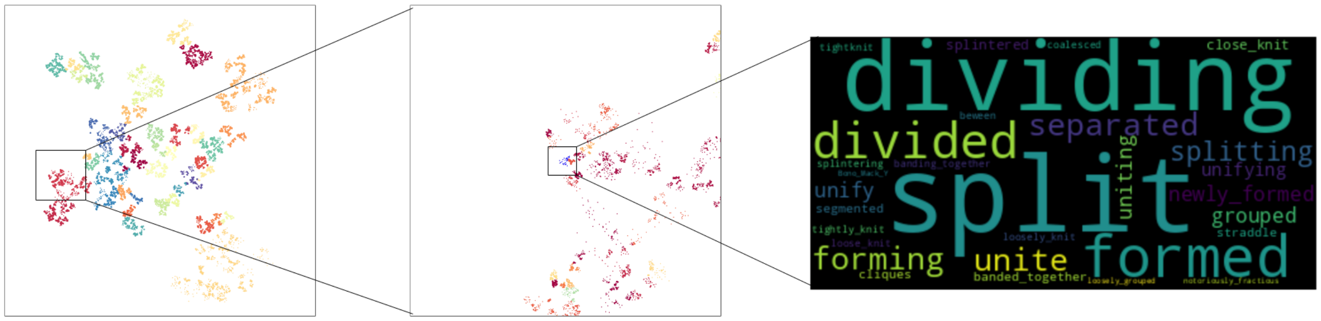


Figure 6. The h-NNE visualization of the unlabelled Google News embeddings. On the left, we see the cluster labels on the top level of the hierarchy. In the middle we zoom into one location and isolate a small cluster using cluster labels from an earlier level of the hierarchy.

structure clusters the data on each level and the final projection is based on those clusterings. This means that any error that might occur in those clusterings is directly translated to reduced quality of the global structure and localization of the points. On the bright side, those classification errors can reflect poor separation of classes in the original space, which is a desired property when one is using h-NNE to inspect the quality of features generated for datasets.

A second limitation is that our method does not preserve by design the topology of the original space. The fact that we use 1-NNs to build our hierarchy implies that any shape homeomorphic to say S^1 will only be preserved if it is contained in a single 1-NN graph component on the lowest level of the tree and if the PCA projection is preserving its circular form. As the size of 1-NNG components is usually quite small this will seldom be the case. In order to remove this limitation one would need to preserve more local properties of the data. This would be possible on higher dimensions, but very low dimensions, like 2 or 3, it conflicts with the preservation of the hierarchical NNG data partitioning which is capturing a more global structure. Therefore, in this case we see this limitation more as a trade-off.

5. Conclusions

We have presented an efficient dimensionality reduction algorithm which utilizes nearest neighbor graph and its hierarchical groupings of points. Using the hierarchy tree nodes as anchor positions in a preliminary projected space, we devise a fast mechanism to move the points directly in the embedding space so that they preserve their local neighborhoods. The embedding process is optimization-free and does not rely on specifying hyperparameters. This results in a demonstrably fast and scalable technique which is shown to preserve both the global and local structure of the data well. The major benefit that our method offers, in addition to its speed and quality, is its ability to expose the clustering structure of the data both in the original and the target space. This may enable analyzing data at different levels of its hierarchical groupings. Faster run-time and the ability to provide clustering labels could be particularly useful for visualizing large-scale unlabelled data.

We expect that our work will enable the analysis of large scale data under reasonable time consumption and hope to trigger interest in the hierarchy-based study of global structural properties of datasets.

References

[1] Ehsan Amid and Manfred K Warmuth. Trimap: Large-scale dimensionality reduction using triplets. *arXiv preprint arXiv:1910.00204*, 2019. 2

[2] El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall, Daniel K Shenfeld, Smita Krishnaswamy, Garry P Nolan, and Dana Pe'er. visne enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature biotechnology*, 31(6):545–552, 2013. 2

[3] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):1–9, 2014. 6

[4] Jacob D. Baron and R. W. R. Darling. K-nearest neighbor approximation via the friend-of-a-friend principle. *arXiv: Combinatorics*, 2019. 6

[5] Martin Bäuml, Makarand Tapaswi, and Rainer Stiefelhagen. Semi-supervised Learning with Constraints for Person Identification in Multimedia Data. In *CVPR*, 2013. 6

[6] Anna C Belkina, Christopher O Ciccolella, Rina Anno, Richard Halpert, Josef Spidlen, and Jennifer E Snyder-Cappione. Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature communications*, 10(1):1–12, 2019. 2

[7] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, sep 1975. 6

[8] Jay Boris. A vectorized "near neighbors" algorithm of order n using a monotonic logical grid. *Journal of Computational Physics*, 66(1):1–20, 1986. 2

[9] Vin De Silva and Joshua B Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *NIPS*, volume 15, pages 705–712, 2002. 6

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[11] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, page 577–586. Association for Computing Machinery, 2011. 6

[12] David Eppstein, Michael S Paterson, and F Frances Yao. On nearest-neighbor graphs. *Discrete & Computational Geometry*, 17(3):263–282, 1997. 2, 3, 4

[13] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 3887–3896. PMLR, 13–18 Jul 2020. 6

[14] Geoffrey Hinton and Sam T Roweis. Stochastic neighbor embedding. In *NIPS*, volume 15, pages 833–840. Citeseer, 2002. 2

[15] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933. 1

[16] Dmitry Kobak and George C Linderman. Initialization is critical for preserving global data structure in both t-sne and umap. *Nature biotechnology*, 39(2):156–157, 2021. 6

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[18] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964. 1

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 1998. 6

[20] George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Efficient algorithms for t-distributed stochastic neighborhood embedding. *arXiv preprint arXiv:1712.09005*, 2017. 2

[21] George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nature methods*, 16(3):243–245, 2019. 2

[22] George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nature methods*, 16(3):243–245, 2019. 7

[23] Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, 2007. 6

[24] Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:824–836, 2020. 6

[25] Leland McInnes. Pynndescent. `https://github.com/lmcinnes/pynndescent`, 2018. 6

[26] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 1, 2, 6, 7

[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 6

[28] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996. 6

[29] Stephen M. Omohundro. Five balltree construction algorithms. Technical report, 1989. 6

[30] Tim Sainburg, Leland McInnes, and Timothy Q Gentner. Parametric umap embeddings for representation and semisupervised learning. *Neural Computation*, 33(11):2881–2907, 2021. 1, 3, 6

[31] John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409, 1969. 1

[32] M. Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR), pages 8934–8943, 2019. 3

[33] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th international conference on world wide web*, pages 287–297, 2016. 1, 2

[34] Laurens Van Der Maaten. Learning a parametric embedding by preserving local structure. In *Artificial Intelligence and Statistics*, pages 384–391. PMLR, 2009. 3

[35] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014. 1, 2, 7

[36] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 1, 2, 7

[37] Jarkko Venna and Samuel Kaski. Local multidimensional scaling. *Neural Networks*, 19(6-7):889–899, 2006. 6

[38] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *Journal of Machine Learning Research*, 22(201):1–73, 2021. 2, 6, 7

[39] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 6