SOFTWARE

openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding

Pavlin G. Poličar[1*], Martin Stražar[1,2] and Blaž Zupan[1,3]

Abstract

Point-based visualizations of large, multi-dimensional data from molecular biology can reveal meaningful clusters. One of the most popular techniques to construct such visualizations is t-distributed stochastic neighbor embedding (t-SNE). Several extensions of t-SNE have recently been proposed to address issues of scalability and the quality of the resulting visualizations. We introduce openTSNE, a modular Python library that implements the core t-SNE algorithm and its extensions. The library is orders of magnitude faster than existing popular implementations, including those from scikit-learn. Unique to openTSNE is also mapping new data to existing embeddings, which can surprisingly assist in solving batch effects.

Keywords: t-SNE; embedding; visualization; dimensionality reduction; Python library

Background

The abundance of high-dimensional data sets in molecular biology calls for dimensionality reduction techniques, particularly for methods that produce informative data visualizations. Popular approaches include principal component analysis (PCA), multidimensional scaling, t-distributed stochastic neighbor embedding (t-SNE) [1], and uniform manifold approximation and projections (UMAP) [2]. Among these, t-SNE lately received much attention as it can address high volumes of data and reveal a meaningful clustering structure. Increased resolution and throughput of modern molecular assays has lead to the frequent use of t-SNE in diverse fields including, but not limited to, single-cell transcriptomics(scRNA-seq, [3, 4, 5]), human genetics [6], metagenomic assembly [7], the spatial organization of microbial communities [8] and metabolomics [9]. Reports on single-cell gene expression data, our running example, often start with an overview of the cell landscape, where t-SNE embeds high-dimensional expression profiles into a two-dimensional space. Figs. 1.a and 1.b show two such embeddings.

Despite its utility, t-SNE has often been criticized for its limited scalability, lack of global organization – t-SNE emphasizes local clusters that are arbitrarily scattered in the low-dimensional space – and the absence of theoretically-founded methods to map new data into existing embeddings [11, 12]. Most of these shortcomings have recently been addressed. Linderman *et al.* sped-up the method through an interpolation-based approximation, achieving linear complexity in the number of samples [13]. Kobak & Berens proposed several techniques to improve global positioning, including estimating similarities with a mixture of Gaussian kernels [14]. In our previous work, we presented a principled approach for embedding new samples into existing visualizations [15].

Results

We introduce openTSNE, a comprehensive Python library that implements t-SNE and all its recently proposed extensions, including:

1. the implementation of efficient approximation schemes [16, 13] allowing the embedding of millions of data points,
2. the addition of new data samples into a fixed existing embeddings [15],
3. improved initialization schemes [17] leading to more globally consistent layouts,
4. variable degrees of freedom [18] allowing the inspection of data at different levels of resolution,
5. multi-scale similarity kernels [14] which preserve small, well-defined clusters and uncover global relationships between clusters, and
6. improved defaults for learning rate and number of iterations [19] which produce embeddings at a lower computational cost.

The openTSNE is compatible with the Python data science ecosystem and libraries that include numpy,

*Correspondence: pavlin.policar@fri.uni-lj.si
[1]Faculty of Computer and Information Science, University of Ljubljana, SI 1000 Ljubljana, Slovenia
Full list of author information is available at the end of the article

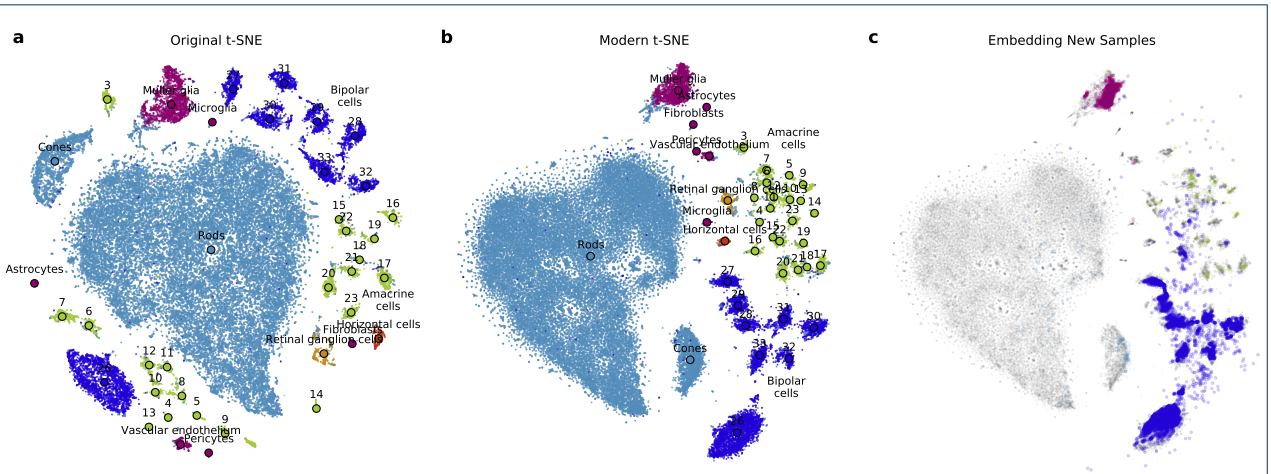

Figure 1 Three t-SNE embeddings demonstrate theoretical advances and corresponding implementations in openTSNE. The data in **(a)** and **(b)** represent 44,808 single-cell gene-expression profiles of mouse retinal cells from Macosko *et al.* [3]. The data in **(c)** contains 27,499 expression profiles from mouse retinal cells from Shekhar *et al.* [10]. **(a)** We construct a t-SNE embedding following the parameter choices from the original publication by Maaten & Hinton [1]. The visualization shows no preservation of the global organization of clusters, resulting from random initialization and an affinity model focused on preserving local neighborhoods. **(b)** A modern t-SNE embedding, utilizing the latest theoretical advances and practical recommendations constructed using a multi-scale affinity model, preserving both short-range and long-range interactions between data points and initialized so that the global layout is meaningful. Unlike in **(a)**, the green and blue clusters representing different sub-types of amacrine and bipolar cells are now localized to the same regions of the space, indicating a higher level of similarity than to other cell types. Embedding in **(c)** shows how existing t-SNE reference atlases can be used to place new samples into existing embeddings. The positions of new data points correspond to cell types from the reference atlas.

scikit-learn, scanpy), thus providing a familiar and intuitive application program interface to the new users. Its modular design encourages extensibility and experimentation with various settings and changes in the analysis pipeline. We have released openTSNE in an open-source to support adaptations by the wider community. At the time of the writing, the library averages several thousands of weekly downloads and has received almost 700 GitHub stars, an appreciation score gained by only 0.3% of public Python repositories. The source code is available at https://github.com/pavlin-policar/openTSNE and supports installations from PyPI and conda-forge, the two most widely adopted Python package managers.

Accessibility of the latest theoretical advances in t-SNE is one of the core design principles of openTSNE. These advances are accessible through a programming interface of the library, where we closely follow the style of the interface as implemented in scikit-learn [20], a popular library for machine learning. The following snippet exemplifies this interface and codes for four visualizations in Fig. 2, demonstrating the effect of a particular recent theoretical advance in the t-SNE algorithm.

```
import openTSNE
from openTSNE.affinity import Multiscale

# a - standard t-SNE
tsne1 = openTSNE.TSNE(perplexity=30).fit(X)
# b - high perplexity  for globa structure
tsne2 = openTSNE.TSNE(perplexity=500).fit(X)
# c - multiscale kernel for local/global structure
aff = Multiscale(X, perplexities=[30, 500])
tsne3 = openTSNE.TSNE(affinities=aff).fit(X)
# d - decrease dof for higher resolution
tsne4 = openTSNE.TSNE(dof=0.6).fit(X)
```

The code snippet considers a numpy array or scipy sparse matrix `X` and creates four `TSNEEmbedding` objects. These objects encode t-SNE embeddings that may subsequently be optimized under different parameter settings or used to embed new samples into the embedding landscape. Function calls may include additional parameters not shown in the example code but explained in accompanying notebooks available at https://github.com/pavlin-policar/opentsne-paper.

Discussion

Uncovering Structure in High-Dimensional Data

Dimensionality reduction techniques implicitly assume that high-dimensional data lies on a lower-dimensional manifold, which can accurately be captured by a smaller number of dimensions. However, there is no evidence that every data set can accurately be described

by using only two dimensions, and any such embedding will therefore inevitably lead to loss of information. It is thus beneficial to examine multiple embeddings, each of which provides a different perspective into topology and other data characteristics.

We illustrate this point by generating four embeddings of the same data on single-cell gene expression in mouse brain [5]. First, Fig. 2.a shows an embedding using default t-SNE parameters. While different clusters of excitatory and inhibitory neurons appear close to one another, all clusters appear equidistant from their neighbors, and the overall relations between groups are not obvious. On the other hand, Fig. 2.b focuses on preserving larger neighborhoods of points, resulting in a more globally consistent layout where relations between clusters become more apparent. In this embedding, it is evident from the increased white space between groups that there is one large class of excitatory neurons, and two related classes of inhibitory neurons. Unfortunately, focusing on preserving large neighborhoods often results in smaller clusters being absorbed into larger ones. Alternatively, Fig. 2.c uses multi-scale similarity kernels which aim to preserve both the global organization of clusters as well as prevent smaller clusters from being absorbed into larger ones. On the other hand, Fig. 2.d shows use the same parameter values as Fig. 2.a, but at a finer level of resolution. This reveals that many of the clusters are composed of other subgroups and may be clustered further into cell subtypes.

When working with data containing millions of data points, standard t-SNE embeddings often become unwieldy – cluster boundaries become blurred, large clusters absorb smaller ones, and relationships between clusters become increasingly difficult to interpret. Fig. 3.a contains expression profiles of over two million single cells captured at different time points in mouse development. The embedding indicates numerous clusters with some transition between time points as indicated by the color-coding, but this is largely difficult to interpret. Kobak & Berens observed that increasing attractive forces between similar data points controlled via the *exaggeration* parameter leads to more compact clusters, and subsequently more informative visualizations [14]. For instance, 3.b doubles the default exaggeration which begins to reveal the overall structure of the data. Doubling the exaggeration again in 3.c, we can now easily observe that the data is comprised of two main groups of cells and eight somewhat smaller clusters. The visualization also reveals a number of tiny clusters, possibly corresponding to rare cell types.

Exaggeration can highlight transitions between cell-states in developmental studies. Standard t-SNE often produces embeddings with clearly defined, discrete clusters. We have shown that we can adjust the level of granularity and resolution of the clusters with several parameters in Fig. 2. However, discrete clusters are often undesired in developmental studies where cells are assumed to follow a continuous trajectory between cell states. To this end, other embedding techniques such as UMAP and ForceAtlas2 [21] tend to better capture the continuity between cell states. Recently, Kobak *et al.* [**?**] have shown that embeddings produced by t-SNE with exaggeration values of 4 and ~ 30 produce embeddings which are markedly similar to UMAP and ForceAtlas2, respectively. For example, in Fig. 3.a, the developmental trajectory between different time points is difficult to observe due to a large number of sprawled out clusters. On the other hand, the development becomes easier to follow when the exaggeration factor is increased from 1 to 4 (Fig. 3.b-c).

Embedding New Samples

Unlike other popular dimensionality reduction methods such as principal component analysis or autoencoders, t-SNE is a non-parametric method and does not define an explicit mapping to the embedding space. Therefore embeddings of new data points need to be found through the use of optimization techniques [15]. openTSNE is currently the only publicly available library allowing users to add new samples to existing embeddings.

Figs. 1.b and 1.c demonstrate how we can use a previously labeled single-cell data set and embed cells from a separate experiment into the reference landscape. The reference data taken from Macosko *et al.* [3] contains gene expression profiles from mouse retinal cells. By embedding the samples from a similar experiment on bipolar retinal cells by Shekhar *et al.* [10], we can correctly map to the bipolar cell clusters in the reference.

Embedding single cells into existing reference atlases can also be useful for cell type classification in cases where cell identities are not known in advance. For instance, in Fig. 4, we construct a reference embedding using labeled data from Hochgerner *et al.* [22] containing gene-expression profiles of cells from the mouse brain. The authors assign a cell type to each cell, and we can verify their accuracy by visualizing well-established gene markers for the major cell types. We then embed cells from Harris *et al.* [23], where labels are provided only for neuronal cells. We quickly identify other non-neuronal cell types, including oligodendrocytes and astrocytes. We use marker genes to validate that new cells were correctly mapped into the reference landscape.

These two examples demonstrate how openTSNE can be used to quickly gain insight into newly-sequenced,

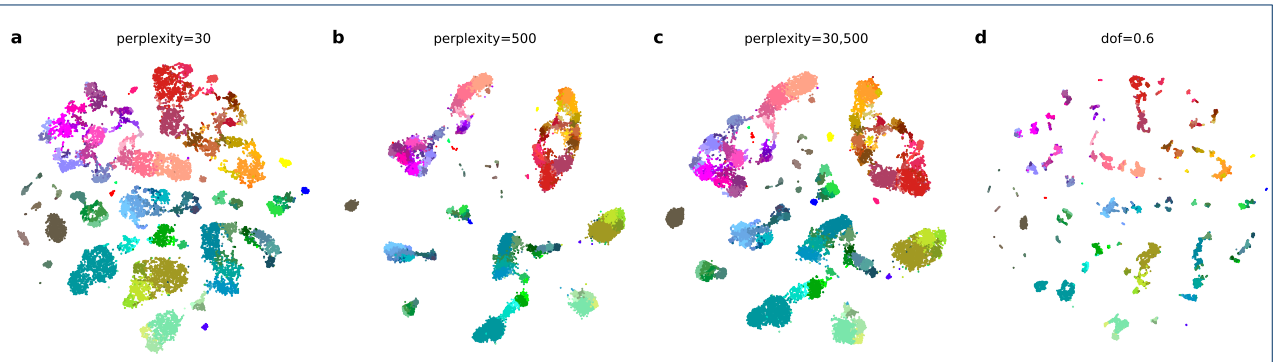

Figure 2 We use openTSNE to create four different visualizations of the Tasic *et al.* [5] data, each providing a different perspective into the topology of the data. The data set contains 21,874 single-cells originating from the mouse neocortex. Cluster annotations and colors are taken from the original publication. Warm colors correspond to excitatory neurons, cool colors correspond to inhibitory neurons, and gray/brown colors correspond to non-neuronal cells. The standard t-SNE **(a)** emphasizes local structure while increasing perplexity **(b)** results in a more meaningful layout of the clusters. We can also combine the two perplexities by using a multiscale kernel affinity model **(c)** and obtain a good trade-off between global and local structure. Alternatively, we can inspect more fine-grained structure and reveal smaller clusters by using a more heavy-tailed kernel **(d)**.

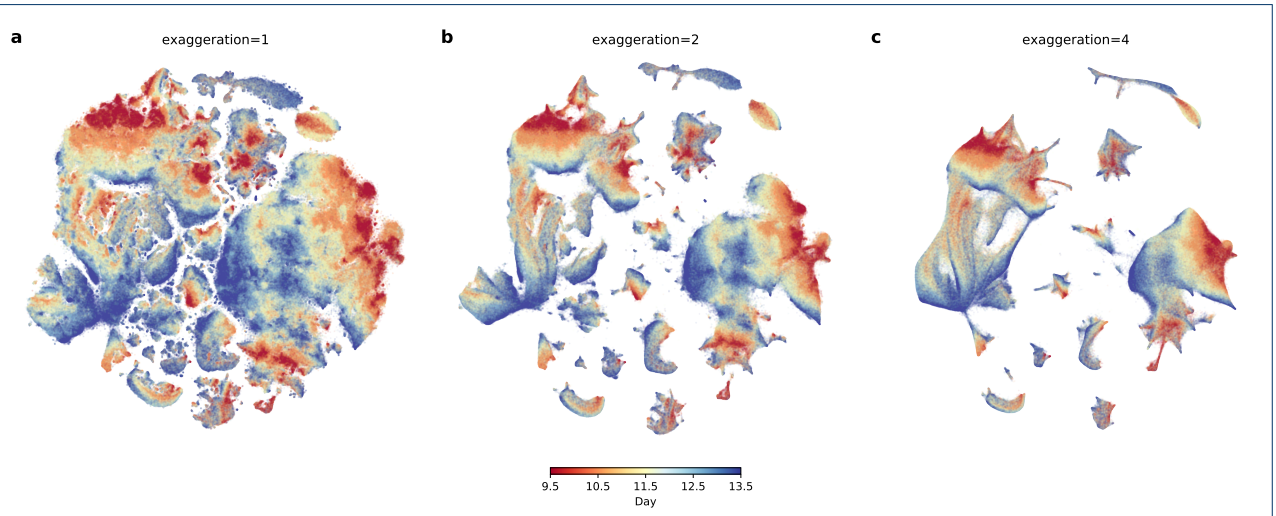

Figure 3 Increasing the exaggeration parameter produces more compact clusters, which highlight the global organization of the clusters and emphasizes continuities between cell states. The data set from Cao *et al.* [4] contains expression profiles from 2,058,652 single cells. The data were collected from mice embryos at different developmental time-points at daily intervals after 9.5 to 13.5 days. **(a)** shows the data is comprised of two main components – the neural tube and mesenchymal cells – as well as several other smaller clusters. Additionally, the colors indicate developmental progression with red indicating least-developed cells and blue indicating most developed cells. The overall developmental trajectory is most apparent with higher levels of exaggeration, showing red cells slowly transitioning into blue cells. Progressively easing the exaggeration factor uncovers finer clusters within the larger groups as shown in **(b)** with exaggeration 2 and subsequently in **(c)**, where we show the standard t-SNE with no exaggeration. 32,011 putative doublets are excluded from the visualizations.

Figure 4 openTSNE enables the user to embed new samples into an existing, reference embedding. We construct a t-SNE embedding for the data from Hochgerner *et al.* [22] containing 24,185 developing, single cells from the mouse hippocampus. The data contains different types of neurons, supporting glia, and other vascular cells (upper left). Cell labels are taken from the original publications. The colors correspond to Fig. 2. We then embed new, hippocampal cells collected by Harris *et al.* [23] using the previous embedding as a reference. In their study, Harris *et al.* collected 6,971 single-cells and focused on identifying different types of inhibitory neurons. However, almost half of the collected cells are not neurons and were left uncharacterized. Inspecting where these cells are embedded in the reference embedding (bottom left) reveals that in addition to inhibitory neurons, the data contains several supporting glial cells as well as a small population of endothelial cells. We can verify the accuracy of our approach by inspecting marker genes for the major cell types in the reference (top row) and embedded samples (bottom row).

single-cell data sets by utilizing existing cell atlases. This approach is general and not limited to single-cell gene expression, but can be used with any tabular data set regardless of field.

Versatility

Versatility is another of openTSNEs core design principles. Kobak & Berens recently introduced several recommendations and tricks to obtain better and more meaningful t-SNE visualizations [14]. These include the use of multi-scale similarity kernels, perplexity annealing, and increasing exaggeration when working with larger data sets. openTSNE provides a flexible API, where these improvements can be incorporated in just a few lines of code. Furthermore, openTSNE supports custom affinity models, enabling users to construct t-SNE embeddings on non-tabular data, e.g. relational data, as the only affinity-model requirement is a notion of similarity between data points, i.e. the distance matrix. Finally, our comprehensive callback system can be utilized to monitor and adapt different stages of the optimization phase and has been used to construct visually appealing animations of the t-SNE optimization process.

Speed

One of t-SNE's common criticisms is limited scalability to large data sets containing up to millions of data points [12]. These criticisms are largely based on two factors. Until quite recently, most popular implementations of t-SNE were based on the Barnes-Hut approximation scheme developed by van der Maaten in 2014 [16] with asymptotic complexity $\mathcal{O}(N \log N)$. More importantly, this criticism was exacerbated further by the fact that the most widely-used implementation of t-SNE came from scikit-learn, which exhibits long runtimes performance compared to its C++ counterpart – MulticoreTSNE. The multi-threaded MulticoreTSNE implementation [24] is perfectly capable of creating t-SNE embeddings of millions of data points in a matter of hours on widely-accessible, consumer-grade processors (Fig. 5). However, recently, Linderman *et al.* developed a new approximation scheme – FIt-SNE – which further reduces the asymptotic time complexity to $\mathcal{O}(N)$, enabling the embedding of large data sets in a matter of minutes.

Fig. 5 benchmarks four Python t-SNE implementations, including those from scikit-learn (v0.23.1), MulticoreTSNE (v0.1), FIt-SNE (v1.1.0), and openTSNE (v0.4.3). We perform benchmarks on two processors, one representing usage on a personal computer and one on a high-performance computing platform. The Intel(R) Core i7 is commonly found in consumer-grade laptop computers, while Intel(R) Xeon(R) processors are found in high-performance computing machines. Benchmarks were run for 1000 iteration with the original t-SNE parameters, as some implementations do not allow their modification.

Fig. 5 demonstrates that both FIt-SNE and openTSNE exhibit better scaling than their Barnes-Hut counterparts – scikit-learn and MulticoreTSNE. Notice that even though FIt-SNE and openTSNE both implement the same approximation scheme, openTSNE tends to be slightly slower. openTSNE is written in Python, which inevitably incurs some runtime overhead compared to its C++ counterpart. However, Fig. 5 shows the runtime of openTSNE is comparable to that of FIt-SNE and much faster than its Barnes-Hut competitors. Additionally, modern computer processors generally contain multiple cores, allowing us to make use of multi-threading. This further reduces the gap between openTSNE and FIt-SNE. On the server-grade processor, openTSNE is slightly faster than its pure C++ counterpart when utilizing multiple cores. We speculate that this is could be because openTSNE uses numpy for most linear algebra operations, which may make better use of the Intel Math Kernel Library (MKL), which is aggressively optimized on Intel(R) Xeon(R) processors.

Additionally, openTSNE provides a flexible API, which allows splitting up the embedding-construction process into several part, caching slow operations. This enables users to quickly experiment with different parameter settings and iterate on their final visualizations.

Ease of Use

Widespread adoption of novel techniques is almost universally accompanied by easy-to-use and easy-to-install software. While the t-SNE implementation from scikit-learn fits this requirement, its implementation is unfortunately very slow, making it unsuitable for its application to large data sets. Other C++ implementations such as MulticoreTSNE and FIt-SNE exhibit better scaling in larger data sets, but do not provide precompiled binaries, requiring users to compile the software themselves. This is problematic from a usability standpoint, as many users often lack the knowledge to do so. This often occurs with Windows operating system, which does not include a C++ compiler, making the correct configuration cumbersome.

openTSNE is a Python package that focuses on accessibility. We provide precompiled binaries for all major Python versions on all major platforms, making the installation process as seamless as possible. openTSNE can be installed through the Python Package Index (PyPI) or through conda on the conda-forge channel. openTSNE is designed to be familiar to Python

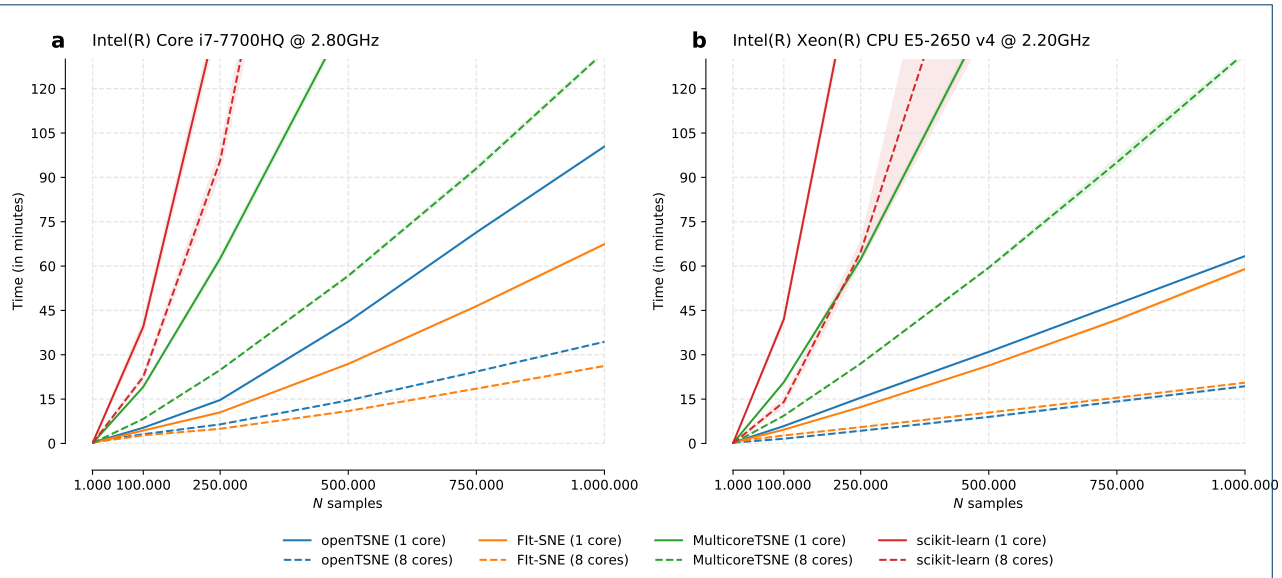

Figure 5 We benchmark openTSNE (v0.4.3) against three popular open-source implementations from scikit-learn [25] (v0.23.1), MulticoreTSNE [24] (v0.1), and FIt-SNE [13] (v1.1.0). Benchmarks were run on a consumer-grade Intel Core i7-7700HQ processor found in laptop computers, and on a server-grade Intel Xeon E5-2650. To generate benchmark data sets of different sizes, we subsampled data from the 10X Genomics 1.3 million mouse brain data set five times, resulting in five different data sets for each size. In total, each implementation was run on 30 different data sets. Notice that openTSNE scales similarly to FIt-SNE, as they both use the same interpolation-based approximation scheme, while scikit-learn and MulticoreTSNE utilize the Barnes-Hut approximation.

users and follows the scikit-learn API [20], which is well established in the Python data science ecosystem. openTSNE implements multi-threaded versions of both the Barnes-Hut and FIt-SNE approximation schemes, allowing it to be used on data sets containing millions of data points. While the Python virtual machine introduces some performance overhead, the runtime is comparable to that of its C++ counterpart – FIt-SNE. Finally, openTSNE is extensible. Its modular design enables researchers to quickly experiment with different settings and easily incorporate their own components into the software. We provide a full feature list and comparison to other popular t-SNE implementations in Table 1.

Conclusion

Text for this section ...

Methods

This section is meant to familiarize the reader with the t-SNE algorithm and its recent extensions. We motivate each extension and introduce standard notation. This brief review is by no means comprehensive, but is self-contained and introduces the mathematics upon which the openTSNE library is based.

Table 1 We compare the feature list of openTSNEto three popular open-source implementations from scikit-learn (v0.23.1), MulticoreTSNE (v0.1), and FIt-SNE (v1.1.0). The first section corresponds to packaging and distribution. A properly packaged library is easily accessible to users and can easily be included as a dependency in other software packages. The second section lists the two approximation schemes. The FIt-SNE approximation scheme is required for t-SNE to scale up to millions of data points. The final section provides a list of extensions and improvements to the standard t-SNE algorithm, many of which can produce markedly better visualizations than under default parameter settings.

	scikit-learn	MulticoreTSNE	FIt-SNE	openTSNE
PyPI package	✓	✓		✓
conda package	✓			✓
Precompiled binaries	✓			✓
Barnes-Hut ($\mathcal{O}(N \log N)$)	✓	✓		✓
FIt-SNE ($\mathcal{O}(N)$)			✓	✓
Multiscale Gaussian kernels			✓	✓
Fully-custom affinity kernels				✓
Variable degrees of freedom			✓	✓
Variable exaggeration			✓	✓
Better initialization			✓	✓
Automatic learning rate			✓	✓
Embedding new samples				✓

Preliminaries

t-distributed stochastic neighbor embedding (t-SNE) is a non-linear dimensionality reduction method, which aims to find a low-dimensional embedding, where local neighborhoods are preserved. More formally, given a multi-dimensional data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \in \mathbb{R}^D$ where N is the number of data points in the data set, t-SNE aims to find a low dimensional embedding $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\} \in \mathbb{R}^d$ where $d \ll D$, such that if points $\mathbf{x}_i$ and $\mathbf{x}_j$ are close in the high-dimensional space, their corresponding embeddings $\mathbf{y}_i$ and $\mathbf{y}_j$ are also close. Since t-SNE is primarily used as a visualization tool, d is typically set to two. The similarity between two data points in the high-dimensional space is defined as:

$$p_{j|i} = \frac{\exp\left(-\frac{1}{2}\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)/\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\frac{1}{2}\mathcal{D}(\mathbf{x}_i, \mathbf{x}_k)/\sigma_i^2\right)}, \quad p_{i|i} = 0 \tag{1}$$

where $\mathcal{D}$ is some distance measure. This is then symmetrized to

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}. \tag{2}$$

The bandwidth of each Gaussian kernel σ_i is selected such that the perplexity of the distribution matches a user-specified parameter value

$$\text{Perplexity} = 2^{H(P_i)} \tag{3}$$

where $H(P_i)$ is the Shannon entropy of P_i,

$$H(P_i) = -\sum_i p_{j|i} \log_2(p_{j|i}). \tag{4}$$

Different bandwidths σ_i enable t-SNE to adapt to the varying density of the data in the multi-dimensional space. Perplexity is sometimes considered the continuous analog to the k nearest neighbors to which distances will be preserved.

The similarity between points $\mathbf{y}_i$ and $\mathbf{y}_j$ in the embedding space is defined using the t-distribution with a single degree of freedom. This is also known as the Cauchy kernel:

$$q_{ij} = \frac{\left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2\right)^{-1}}, \quad q_{ii} = 0. \tag{5}$$

Finally, the Kullback-Leibler (KL) divergence is used as a measure of agreement between distributions $\mathbf{P}$ and $\mathbf{Q}$

$$C = \text{KL}(\mathbf{P} \mid\mid \mathbf{Q}) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{6}$$

The t-SNE objective is to find embeddings $\mathbf{Y}$ such that the KL divergence is minimized.

The corresponding gradient takes the form

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) (\mathbf{y}_i - \mathbf{y}_j) \left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2\right)^{-1}. \tag{7}$$

Optimization is performed using batch gradient descent with the delta-bar-delta update rule [26]. Originally, t-SNE was run for 1000 iterations consisting of two phases. In the first *early exaggeration* phase, the attractive forces between data points is increased by a factor $\rho = 12$ so that points can more easily move throughout the embedding to find their respective neighbors. The remaining 750 iterations are run with $\rho = 1$, which reverts the attractive forces to their original values and produces the final embedding.

Belkina *et al.* later found that faster convergence can be achieved by increasing the learning rate to $\eta = N/12$ [19]. The number of iterations can then safely be lowered to 750, leading to faster runtime. This convention has been adopted by most modern t-SNE implementations.

Efficient Approximation Schemes

A direct evaluation of t-SNE gradients requires $\mathcal{O}(N^2)$ operations, which makes its application impractical to any reasonably-sized data set and beckons for the development of efficient approximation schemes. Van der Maaten observed that the t-SNE gradient may be reformulated in terms of a N-body problem where data points may be interpreted as particles which attract and repel each other [16]. The gradient may be rewritten as

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \left[\sum_{j \neq i} p_{ij} q_{ij} Z (\mathbf{y}_i - \mathbf{y}_j) - \sum_{j \neq i} q_{ij}^2 Z (\mathbf{y}_i - \mathbf{y}_j) \right], \tag{8}$$

where $Z = \sum_{k \neq l} \left(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2\right)^{-1}$. The two terms may be viewed as the attractive and repulsive forces between particles, respectively. Each term lends itself to efficient approximations, which enable us to greatly reduce the time complexity of t-SNE.

Attractive Forces

Van der Maaten observed that since p_{ij} is calculated in the high-dimensional space using a Gaussian kernel and that the bandwidth is selected such that only a fixed number of closest neighbors (determined by the perplexity parameter) will contribute to the attractive forces of a given point due to the exponential decay of the Gaussian kernel [16]. Therefore it is sufficient to calculate the attractive forces only over a relatively small number of nearest neighbors instead of all N points. By utilizing tree-based nearest-neighbor search methods, the time complexity is thus reduced to $\mathcal{O}(N \log N)$. Linderman *et al.* further realized that, qualitatively, embeddings are visually virtually indistinguishable when using only *approximate* nearest neighbors, further reducing time complexity to $\mathcal{O}(N)$ [13].

Repulsive Forces

Examining the second term of Eqn. (8) we notice that each point indiscriminately exerts a repulsive force on all other points. Van der Maaten proposed an approach based on N-body simulations and used a space-partitioning Barnes-Hut tree approach to approximate the interaction between data points [16]. Briefly, the approach splits the space into quadrants (in the 2D case) and if a given point is far enough away from a quadrant, the points in that quadrant are summarized by a single center of mass. This reduces the time complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$.

More recently, Linderman *et al.* proposed an alternative approach, FIt-SNE, based on non-uniform convolutions to calculating all pairwise interactions between repelling data points [13]. Briefly, Linderman *et al.* observed that the repulsive forces $\mathbf{R}$ may be rewritten as

$$\begin{aligned}\mathbf{R}_i &= \sum_{j \neq i} q_{ij}^2 Z \left(\mathbf{y}_i - \mathbf{y}_j\right) \\ &= \sum_{j \neq i} \frac{\mathbf{y}_i - \mathbf{y}_j}{\left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2\right)^2} \Big/ \sum_{k \neq l} \frac{1}{1 + ||\mathbf{y}_k - \mathbf{y}_l||^2}\end{aligned} \tag{9}$$

and computed by evaluating three terms

$$\begin{aligned}\phi_{1,j} &= \sum_{j \neq i} \frac{1}{1 + ||\mathbf{y}_j - \mathbf{y}_i||^2}, \\ \phi_{2,j} &= \sum_{j \neq i} \frac{\mathbf{y}_j}{\left(1 + ||\mathbf{y}_j - \mathbf{y}_i||^2\right)^2}, \\ \phi_{3,j} &= \sum_{j \neq i} \frac{1}{\left(1 + ||\mathbf{y}_j - \mathbf{y}_i||^2\right)^2}.\end{aligned} \tag{10}$$

Then the numerator and denominator of Eqn. (9) can be computed as $\mathbf{y}_i \phi_{1,j} - \phi_{2,j}$ and $Z = \sum_j \phi_{3,j}$, respectively. These interactions are accelerated by interpolating these terms through a grid of equispaced interpolation points. This shifts the computational burden onto the interpolation points and reduces the time complexity to $\mathcal{O}(N)$.

openTSNE provides efficient implementations for all the aforementioned improvements but defaults to using approximate nearest neighbor search and the non-uniform convolutions approach as these enable the user to quickly create informative visualizations for data sets containing up to millions of samples.

Embedding New Samples

t-SNE is non-parametric and does not define an explicit mapping from the high-dimensional space to the embedding space. Therefore embeddings of new data points need to be found through the use of optimization techniques [15]. When adding new data points to an existing, reference embedding, the reference data points are fixed in place and the new data points are allowed to find their respective positions. The optimization remains the same as in standard t-SNE with only slight modifications to p_{ij} and q_{ij}

$$p_{j|i} = \frac{\exp\left(-\frac{1}{2}\mathcal{D}(\mathbf{x}_i, \mathbf{v}_j)/\sigma_i^2\right)}{\sum_i \exp\left(-\frac{1}{2}\mathcal{D}(\mathbf{x}_i, \mathbf{v}_j)/\sigma_i^2\right)}, \tag{11}$$

$$q_{j|i} = \frac{\left(1 + ||\mathbf{y}_i - \mathbf{w}_j||^2\right)^{-1}}{\sum_i \left(1 + ||\mathbf{y}_i - \mathbf{w}_j||^2\right)^{-1}}, \tag{12}$$

where $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M\} \in \mathbb{R}^D$ where M is the number of samples in the secondary data set and $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M\} \in \mathbb{R}^d$. Additionally, we omit the symmetrization step in Eqn. (2). The gradients of $\mathbf{w}_j$ with respect to the loss (Eqn. 6) are:

$$\frac{\partial C}{\partial \mathbf{w}_j} = 2 \sum_i \left(p_{j|i} - q_{j|i}\right)\left(\mathbf{y}_i - \mathbf{w}_j\right)\left(1 + ||\mathbf{y}_i - \mathbf{w}_j||^2\right)^{-1} \tag{13}$$

As in standard t-SNE, a direct calculation of gradients takes $\mathcal{O}(N \cdot M)$ time, but it is straightforward to adapt the Barnes-Hut and FIt-SNE approximation schemes, reducing the time complexity to $\mathcal{O}(M \log N)$ and $\mathcal{O}(\max\{N, M\})$, respectively. In the FIt-SNE approximation scheme, we additionally exploit the fact that the reference embedding remains fixed throughout the optimization of newly added points, and precompute the interpolation grid. This further reduces the runtime complexity from $\mathcal{O}(\max\{N, M\})$ to $\mathcal{O}(M)$.

Alternative Perplexity Kernels

In standard t-SNE, distances are converted to similarities through the use of Gaussian kernels of varying bandwidths. The bandwidths are indirectly determined by the user-specified perplexity parameter so that a fixed number of nearest data points will have non-zero similarity. One common trick for uncovering the global relations between clusters is to increase perplexity so that more long-range interactions are preserved in the final embedding. However, one unfortunate side effect of increasing perplexity is that smaller clusters get absorbed into larger ones.

Kobak & Berens [14] suggest that replacing the Gaussian kernel with a mixture of Gaussians may provide better insight into both the local and global structure. Therefore, the similarities between data points in the input space become

$$p_{j|i} \propto \frac{1}{\sigma_{1,i}} \exp\left(-\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)/2\sigma_{1,i}^2\right) + \frac{1}{\sigma_{2,i}} \exp\left(-\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)/2\sigma_{2,i}^2\right). \tag{14}$$

The bandwidths of each Gaussian $\sigma_{1,i}$ and $\sigma_{2,i}$ is determined by selecting different perplexity values. They show that using a kernel with perplexity 500 captures the long-range interactions which preserve global cluster organization in the final embedding and combine this with a kernel with perplexity 50, which prevents small, well-defined clusters from being absorbed into larger ones.

Variable Degrees of Freedom

Standard t-SNE reveals the clustering structure in data at one level of resolution. Different perplexity parameter values can be used to identify global cluster relationships or small, well-isolated groups. Unfortunately, varying perplexity values can be time-consuming as this involves recomputing the KNNG which is often the most expensive part of the t-SNE algorithm. Alternatively, Kobak *et al.* [18] suggest that varying the degree of freedom in the t-distribution used in the embedding space during optimization can be used to explore the clustering structure at different levels of resolution.

Standard t-SNE models similarities between data points in the embedding space using a t-distribution with a single degree of freedom

$$q_{ij} \propto \left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2/\alpha\right)^{-\alpha} = \frac{1}{\left(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2/\alpha\right)^{\alpha}}. \tag{15}$$

In standard t-SNE $\alpha = 1$ so this simplifies to the Cauchy kernel from Eqn. (5).

The gradient of the loss function then becomes

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} \left(p_{ij} - q_{ij}\right) q_{ij}^{1/\alpha} \left(\mathbf{y}_i - \mathbf{y}_j\right), \tag{16}$$

which can, again, be decomposed into the attractive and repulsive forces as

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} p_{ij} q_{ij}^{1/\alpha} (\mathbf{y}_i - \mathbf{y}_j) - 4 \sum_{j \neq i} q_{ij}^{\frac{\alpha+1}{\alpha}} / Z (\mathbf{y}_i - \mathbf{y}_j). \tag{17}$$

Adapting existing approximation schemes to this formulation is straightforward. We provide efficient implementations of both the Barnes-Hut and the FIt-SNE approximation schemes, where we replace the terms from Eqn. (10) with

$$\phi_{1,j} = \sum_{j \neq i} \frac{1}{\left(1 + ||\mathbf{y}_j - \mathbf{y}_i||^2/\alpha\right)^{\alpha+1}},$$
$$\phi_{2,j} = \sum_{j \neq i} \frac{\mathbf{y}_j}{\left(1 + ||\mathbf{y}_j - \mathbf{y}_i||^2/\alpha\right)^{\alpha+1}},$$
$$\phi_{3,j} = \sum_{j \neq i} \frac{1}{\left(1 + ||\mathbf{y}_j - \mathbf{y}_i||^2/\alpha\right)}.$$

Variable Exaggeration

Embeddings produced by standard t-SNE often produce clusters separated by thin boundaries and make use of all available space. While this is often desirable, this often obscures the global relationships between clusters as all neighboring clusters appear at the same distance from one another. Other dimensionality reduction methods such as UMAP [2] or ForceAtlas2 [21] tend to produce embeddings where clusters appear more compact and the white-space separating the clusters may be interpreted at least partially as a loose measure of distance.

Kobak *et al.* [**?**] showed that the early exaggeration factor ρ in t-SNE may be increased to produce more compact layouts often seen in UMAP or ForceAtlas2. Early exaggeration is a trick introduced by van der Maaten [1] used in the first 250 iterations and increases the attractive forces between data points, enabling neighboring points to move close to one another while ignoring the repulsive forces between data points. In standard t-SNE, the remainder of the iterations are run with $\rho = 1$. Kobak *et al.* showed that using $\rho = 4$ and $\rho = 30$ in the second phase of the optimization

results in embeddings visually similar to UMAP embeddings, and ForceAtlas2 embeddings, respectively.

While it is difficult to claim one is better than the other, different settings of ρ may uncover different properties and clustering structures. For example, in single-cell data, standard t-SNE with $\rho = 1$ often uncovers distinct groups of cell-types but obscures developmental paths. This problem is exacerbated when dealing with large numbers of data points. Conversely, ForceAtlas2 has often been used to uncover trajectories and transitions between cell types, but larger clusters often absorb small, distinct groups of cells.

Globally Consistent Initialization Schemes

Similarly to UMAP and ForceAtlas2, t-SNE can also be viewed as a force-directed layout algorithm. The first phase of the t-SNE algorithm constructs a k-nearest neighbor graph, and the subsequent optimization balances the attractive and repulsive forces between data points.

The final positioning of data points using this class of algorithms is largely dependent on the initial positions in the embedding space. UMAP is often said to better preserve global structure than t-SNE, but Kobak & Linderman have recently discovered that this supposed advantage is entirely due to the better default initialization in UMAP than in older implementations of t-SNE [17]. Standard implementations of t-SNE use random initializations, which often scatter resulting clusters throughout the embedding space. UMAP injects global organization into the embedding by initializing point positions using Laplacian eigenmaps. When initialized in the same manner, both t-SNE and UMAP perform equally well, or equally poorly, dispelling the myth that UMAP is superior in this respect.

openTSNE defaults to using the two leading principal components as initialization, but also provides a spectral approach, similar to that of UMAP. The clusters in resulting embeddings are organized in a globally coherent manner, leading to more interpretable visualizations.

Availability

openTSNE is distributed under the BSD-3-Clause License and is publicly available as an open-source package at https://github.com/pavlin-policar/openTSNE. openTSNE is also available on PyPI and conda-forge. The data sets used and scripts used in this study are included in accompanying notebooks, publicly available at https://github.com/pavlin-policar/opentsne-paper.

Competing interests
The authors declare that they have no competing interests.

Author's contributions
Text for this section . . .

Acknowledgements
We would like to thank Dmitry Kobak for various helpful discussions and best practices when using t-SNE, as well as his contributions to the source code. We would also like to thank George Linderman for his help with the FIt-SNE algorithm.

Author details
[1]Faculty of Computer and Information Science, University of Ljubljana, SI 1000 Ljubljana, Slovenia. [2]Broad Institute of Harvard and MIT, MA 02142 Cambridge, U.S.A.. [3]Department of Molecular and Human Genetics, Baylor College of Medicine, TX 77030 Houston, U.S.A..

References

1. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(Nov), 2579–2605 (2008)
2. McInnes, L., Healy, J., Melville, J.: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv e-prints (2018). 1802.03426
3. Macosko, E.Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., Tirosh, I., Bialas, A.R., Kamitaki, N., Martersteck, E.M., *et al.*: Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell **161**(5), 1202–1214 (2015)
4. Cao, J., Spielmann, M., Qiu, X., Huang, X., Ibrahim, D.M., Hill, A.J., Zhang, F., Mundlos, S., Christiansen, L., Steemers, F.J., *et al.*: The single-cell transcriptional landscape of mammalian organogenesis. Nature **566**(7745), 496–502 (2019)
5. Tasic, B., Yao, Z., Graybuck, L.T., Smith, K.A., Nguyen, T.N., Bertagnolli, D., Goldy, J., Garren, E., Economo, M.N., Viswanathan, S., *et al.*: Shared and distinct transcriptomic cell types across neocortical areas. Nature **563**(7729), 72–78 (2018)
6. Hirata, J., Hosomichi, K., Sakaue, S., Kanai, M., Nakaoka, H., Ishigaki, K., Suzuki, K., Akiyama, M., Kishikawa, T., Ogawa, K., *et al.*: Genetic and phenotypic landscape of the major histocompatibilty complex region in the japanese population. Nature genetics **51**(3), 470–480 (2019)
7. Beaulaurier, J., Zhu, S., Deikus, G., Mogno, I., Zhang, X.-S., Davis-Richardson, A., Canepa, R., Triplett, E.W., Faith, J.J., Sebra, R., *et al.*: Metagenomic binning and association of plasmids with bacterial host genomes using dna methylation. Nature biotechnology **36**(1), 61 (2018)
8. Sheth, R.U., Li, M., Jiang, W., Sims, P.A., Leong, K.W., Wang, H.H.: Spatial metagenomic characterization of microbial biogeography in the gut. Nature biotechnology **37**(8), 877–883 (2019)
9. Tkachev, A., Stepanova, V., Zhang, L., Khrameeva, E., Zubkov, D., Giavalisco, P., Khaitovich, P.: Differences in lipidome and metabolome organization of prefrontal cortex among human populations. Scientific reports **9**(1), 1–10 (2019)
10. Shekhar, K., Lapan, S.W., Whitney, I.E., Tran, N.M., Macosko, E.Z., Kowalczyk, M., Adiconis, X., Levin, J.Z., Nemesh, J., Goldman, M., *et al.*: Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. Cell **166**(5), 1308–1323 (2016)
11. Ding, J., Condon, A., Shah, S.P.: Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. Nature Communications **9**(1), 2002 (2018)
12. Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I.W., Ng, L.G., Ginhoux, F., Newell, E.W.: Dimensionality reduction for visualizing single-cell data using umap. Nature Biotechnology **37**(1), 38 (2019)
13. Linderman, G.C., Rachh, M., Hoskins, J.G., Steinerberger, S., Kluger, Y.: Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. Nature Methods **16**(3), 243–245 (2019)
14. Kobak, D., Berens, P.: The art of using t-sne for single-cell transcriptomics. Nature Communications **10**(1), 1–14 (2019)
15. Poličar, P.G., Stražar, M., Zupan, B.: Embedding to reference t-sne space addresses batch effects in single-cell classification. In: International Conference on Discovery Science, pp. 246–260 (2019). Springer
16. Van Der Maaten, L.: Accelerating t-sne using tree-based algorithms. The Journal of Machine Learning Research **15**(1), 3221–3245 (2014)

17. Kobak, D., Linderman, G.C.: Umap does not preserve global structure any better than t-sne when using the same initialization. bioRxiv (2019)
18. Kobak, D., Linderman, G., Steinerberger, S., Kluger, Y., Berens, P.: Heavy-tailed kernels reveal a finer cluster structure in t-sne visualisations. arXiv preprint arXiv:1902.05804 (2019)
19. Belkina, A.C., Ciccolella, C.O., Anno, R., Halpert, R., Spidlen, J., Snyder-Cappione, J.E.: Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. Nature Communications **10**(1), 1–12 (2019)
20. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013)
21. Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. PloS one **9**(6), 98679 (2014)
22. Hochgerner, H., Zeisel, A., Lönnerberg, P., Linnarsson, S.: Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell rna sequencing. Nature Neuroscience **21**(2), 290–299 (2018)
23. Harris, K.D., Hochgerner, H., Skene, N.G., Magno, L., Katona, L., Gonzales, C.B., Somogyi, P., Kessaris, N., Linnarsson, S., Hjerling-Leffler, J.: Classes and continua of hippocampal ca1 inhibitory neurons revealed by single-cell transcriptomics. PLoS Biology **16**(6), 2006387 (2018)
24. Ulyanov, D.: Multicore-TSNE. GitHub (2016)
25. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., *et al.*: Scikit-learn: Machine learning in python. the Journal of Machine Learning Research **12**, 2825–2830 (2011)
26. Jacobs, R.A.: Increased rates of convergence through learning rate adaptation. Neural Networks **1**(4), 295–307 (1988)