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 visualisations of large, multi-dimensional data from molecular biology can reveal meaningful clusters. One of the most popular techniques to construct such visualisations is t-distributed stochastic neighbor embedding (t-SNE), for which a number of extensions have recently been proposed to address issues of scalability and the quality of the resulting visualisations. 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 the mapping of new data to existing embeddings, which can surprisingly assist in solving batch effects.

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

Background

The abundance of high-dimensional data sets in molecular biology calls for techniques for dimensionality reduction, and in particular for methods that can help in the construction of data visualizations. Popular approaches for dimensionality reduction 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 meaningful clustering structure. Most of the recent reports on single-cell gene expression data start with an overview of the cell landscape, where t-SNE embeds high-dimensional expression profiles into a two-dimensional space [3, 4, 5]. Figs. 1.a and 1.b are two examples of such embeddings.

Despite its utility, t-SNE has often been criticized for poor scalability when addressing large data sets, lack of global organization – t-SNE focuses on local clusters that are arbitrarily scattered in the low-dimensional space – and absence of theoretically-founded implementations to map new data into existing embeddings [7, 8]. Most of these shortcomings, have recently been addressed. Linderman *et al.* sped-up the method through an interpolation-based approximation, reducing the time complexity to be merely linear dependent on the number of samples [9]. Kobak *et al.* proposed several techniques to improve global positioning, including estimating similarities with a mixture of Gaussian kernels [10]. In our previous work, we presented a principled approach for embedding new samples into existing visualizations [11].

Results

We introduce openTSNE, a comprehensive Python library that implements t-SNE and all its recently proposed extensions. Most notable among these are the implementation of efficient approximation schemes [12, 9] allowing the embedding of millions of data points, the addition of new samples into existing t-SNE embeddings [11], better initialization schemes [13] leading to more globally consistent layouts, variable degrees of freedom [14] allowing the inspection of data at different levels of resolution, multiscale similarity kernels [10] which emphasize the preservation of both short-range and long-range interactions, and improved default parameters [15] which produce better embeddings at lower computational cost. The library is compatible with the Python data science ecosystem (e.g., numpy, scikit-learn, scanpy) providing a familiar and intuitive user interface to new users. Its modular design encourages extensibility and experimentation with various settings and changes in the analysis pipeline.

openTSNE is open-source and has become an integral part of the Python data-science ecosystem. The package has received over 600 Github stars, and has been downloaded hundreds of thousands of times, averaging

*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

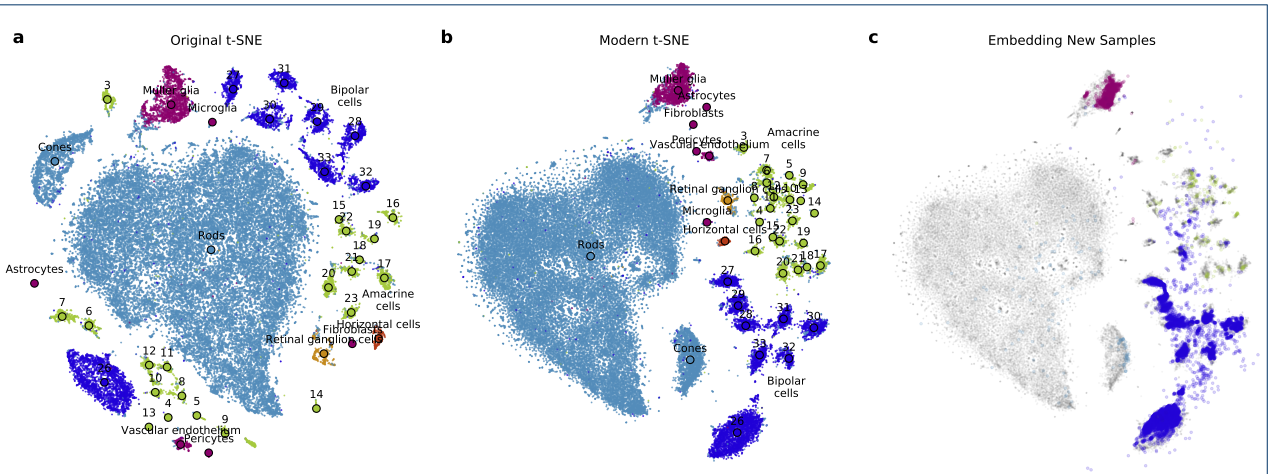

Figure 1 The figure shows the progression of t-SNE embeddings following theoretical advances, which are all included in openTSNE. The data in **(a)** and **(b)** represent 44,808 single-cell gene-expression profiles of mouse retinal cells from the study by Macosko *et al.* [3]. Similarly, the data in **(c)** contains 27,499 expression profiles from mouse retinal cells from Shekhar *et al.* [6]. **(a)** We construct a t-SNE embedding following the parameter choices from the original publication from Maaten & Hinton [1]. The visualization shows poor preservation of the global organisation of clusters, resulting from random initialization and an affinity model focused on preserving local neighborhoods. **(b)** shows a modern t-SNE embedding, utilizing the latest theoretical advances and practical recommendations. The embedding is constructed using a multi-scale affinity model, preserving both short-range and long-range interactions between data points, and is initialized such 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. **(c)** shows how existing t-SNE reference atlases can be used to embed new samples into existing embeddings. The positions of new data points correspond to cell types from the reference atlas.

in thousands of weekly downloads. The source code is publicly available on Github at `https://github.com/pavlin-policar/openTSNE`, and can be installed from PyPI and conda-forge – the two most widely adopted Python package managers.

Accessibility is one of the core design principles of openTSNE. Our aim is to make the latest theoretical advances in t-SNE as accessible as possible to as many users as possible. As such, our API is based on the familiar scikit-learn interface [16], enabling a smooth transition from other Python data science libraries. For example, Fig. 2 depicts four different t-SNE embeddings, each based on recent advances in t-SNE visualizations. The code used to generate these embeddings is listed below.

```
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 takes as input a numpy array or scipy sparse matrix `X` and produces four `TSNEEmbedding` objects. These may subsequently be subject to further optimization under different parameter settings, or may be used to embed new samples into the embedding landscape. Additional parameters are omitted for clarity, but can be found in accompanying notebooks publicly available at `https://github.com/pavlin-policar/opentsne-paper`.

Discussion

Uncovering Structure in High-Dimensional Data Sets

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 to suggest that any data set can accurately be described using only two dimensions, and that any projection onto a two-dimensional plane will inevitably lead to loss of information. Therefore, it is beneficial to examine multiple embeddings, each of which provide a different perspective into the topology and other characteristics of the data. For instance, in Fig. 2, we show four embeddings of the same gene-expression data from the mouse brain [5].

Fig. 2.a shows a t-SNE embedding using default parameters, which clearly uncovers numerous clusters.

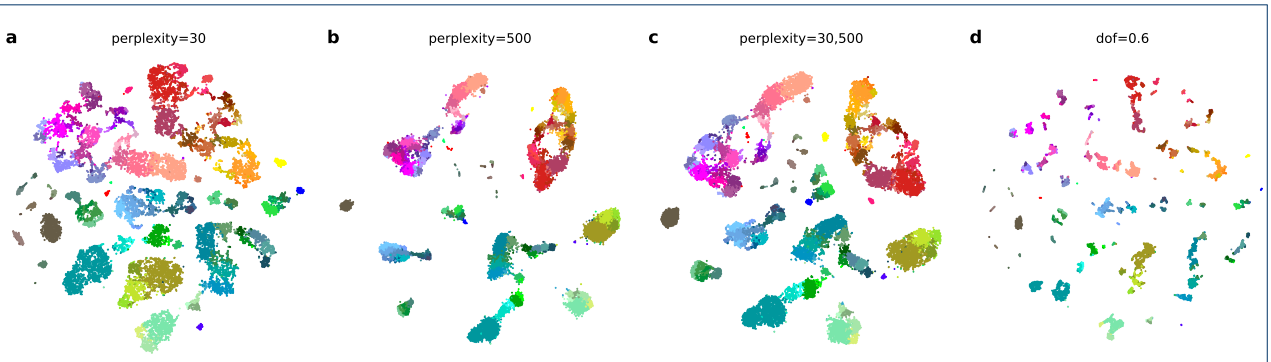

Figure 2 We use openTSNE to create four different visualizations of the Tasic *et al.* [5] data set containing 21,874 single-cells originating from the mouse neocortex. Cluster annotations and colors are taken from the original publication. Warm colors correspond to excitatory neurons, cool colors correspond 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)**.

Warm colors correspond to excitatory neurons, while cool colors represent inhibitory neurons. While clusters of excitatory and inhibitory neurons appear close together, all clusters appear somewhat equidistant from their neighobrs, and the overall relations between clusters are hidden from the user. On the other hand, Fig. 2.b focuses on preserving larger neighborhoods of points, resulting in a much more globally consistent layouts where relations between clusters become more apparent. In this embedding, it is evident that there is one large group of excitatory neurons, and two related groups 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 organisation of clusters as well as prevent smaller clusters from being obscured by larger ones. Fig. 2.d shows another lever available to users used to uncover clustering structure at different levels of resolution. Kobak *et al.* [14] found that decreasing the degrees of freedom in the t-distribution an reveal smaller, more compact clusters, otherwise easily missed in other embeddings.

When faced with a data set containing millions of data points, standard t-SNE embeddings often become unwieldy, with blurred cluster boundaries, large clusters absorbing smaller ones, and relationships between clusters becoming increasingly difficult to interpret. Fig. 3.a contains expression profiles of 2,058,652 single cells captured at different time points in mouse development. The embedding clearly indicates numerous clusters with some transition between time points, but this is largely difficult to reason about. Kobak & Berens observed that increasing attractive forces between similar data points controlled via the *exaggeration* parameter leads to more compact clusters, which leads to more informative visualizations [10]. For instance, 3.b increases the doubles the default exaggeration rate and 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 large clusters, and eight smaller ones. Coincidentally, embeddings with this level of exaggeration roughly correspond to embeddings produced by UMAP [**?**].

Exaggeration can be used to highlight transitions between cell-states in developmental studies. Standard t-SNE often produces embeddings with clearly-defined, discrete clusters. We have shown that users have access to several parameters to control the granularity of the clustering structure. However, such properties are undesired in developmental studies as we assume that the cells follow a continuous path between cell states. To this end, other embedding techniques such as UMAP and ForceAtlas2 are often used, as they 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 visually very similar to UMAP and ForceAtlas2. For example, in Fis. 3.a the colors clearly indicate some transitional states, which are made much more apparent in Fig. 3.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

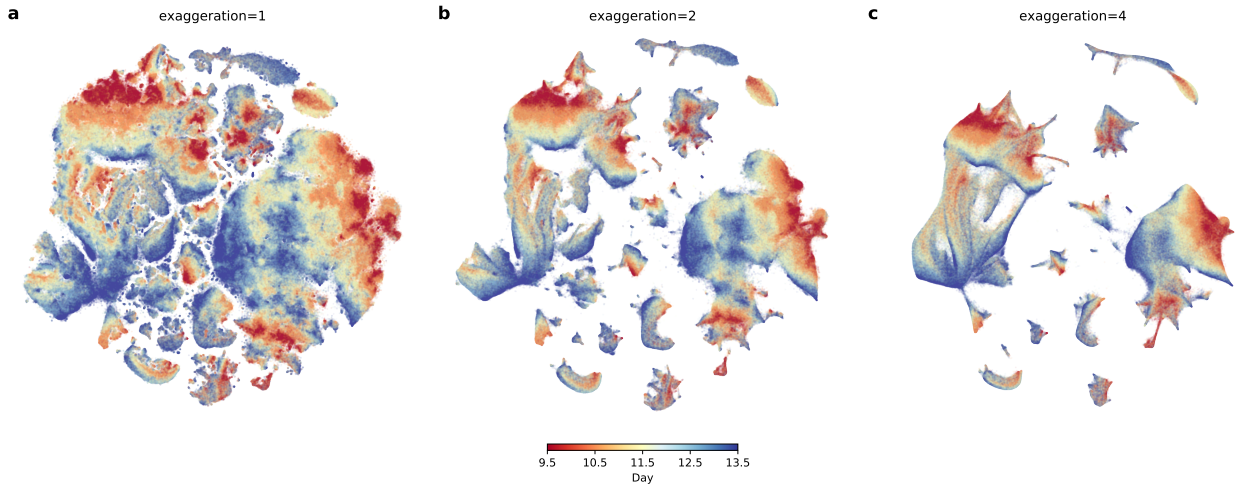

Figure 3 The data set from Cao *et al.* [4] contains expression profiles from 2,058,652 single cells. The data were collected from mice embryos at different developmental time-points at daily intervals after 9.5 to 13.5 days. The exaggeration parameter can be used to highlight the global organization of the clusters and enforce cluster compactness. **(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.

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 [11]. 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 annotated single-cell data set and embed data from a different single-cell experiment into the reference landscape. The reference data taken from Macosko *et al.* [3] contains gene-expression profiles from mouse retinal cells. Taking data from and experiment focusing only on bipolar retinal cells by Shekhar *et al.* and embedding these cells into the reference embedding, we can see that the majority of new cells 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 yet known. For instance, in Fig. 4, we construct a reference embedding using data from Hochgerner *et al.* [17] containing gene-expression profiles of cells from the mouse brain. The authors provide cell annotations for each cell, and we can verify their accuracy by visualizing well-established gene markers for the major cell types. We can then embed similar data from Harris *et al.*, where annotations are provided only for neuronal cells, and quickly identify other non-neuronal cell types, including oligodendrocytes and astrocytes. We then use the 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, single-cell data sets by utilizing already existing, publicly-available, reference atlases. This approach is extremely general and is not limited to single-cell gene-expression data, but can be applied to any tabular data.

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 embeddings [10]. These include the use of multi-scale similarity kernels, slowly easing the perplexity parameter, and increasing exaggeration when working with larger data sets. openTSNE provides an extremely flexible API, where these improvements can be incorporated in a few lines of code. Furthermore, openTSNE additionally supports custom affinity models, e.g. multi-scale Gaussian mixtures or uniform kernels, allowing for quick experimentation. This also enables users to construct t-SNE embeddings on non-tabular data, e.g. relational data, as the only

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.* [17] 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 annotations are taken from the original publications. The colors correspond to Fig. 2. We then embed new, hippocampal cells collected by Harris *et al.* [18] 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 a number of supporting glia 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).

affinity-model requirement is some notion of similarity between data points. Finally, our comprehensive callback system can 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 poor scalability to large data sets containing up to millions of data points [8]. 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 [12] with asymptotic complexity $\mathcal{O}(N \log N)$. More importantly, this misconception was exacerbated further by the fact that the most widely-used implementation of t-SNE came from scikit-learn, which exhibits very poor runtime performance compared to its C++ counterpart – MulticoreTSNE. Even with the Barnes-Hut approximation, the multi-threaded MulticoreTSNE implementation [19] is perfectly capable of creating t-SNE embeddings of millions of data points in a matter of hours on widely-accessible, consumer-grade CPUs (Fig. 5). Recently, Linderman *et al.* developed a new approximation scheme – FIt-SNE – which further reduces the asymptotic time complexity to $\mathcal{O}(N)$, embedding large data sets in a matter of minutes.

Fig. 5 shows benchmarks of four Python t-SNE implementations, including 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 often found on high-performance computing machines. Benchmarks were run with the original t-SNE parameters for 1000 iterations, as some implementations do not enable their modification.

Fig. 5 demonstrates that both FIt-SNE and openTSNE exhibit much better runtime performance 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 introduces some runtime overhead compared to its pure C++ counterpart. However, Fig. 5 demonstrates 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 more than a single core, allowing us to make use of multithreading, which drastically reduces the runtime differences between openTSNE and FIt-SNE. On the server-grade processor, openTSNE is actually slightly faster than its pure C++ counterpart. We speculate that this is likely due to the higher level API openTSNE uses for linear algebra operations, which may make better use of the Intel Math Kernel Library (MKL), which are aggressively optimized on Xeon processors.

Additionally, openTSNE provides a flexible API, which allows splitting up the embedding-construction process and caching slow operations, enabling the user to quickly experiment with different parameter settings.

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 in scikit-learn fits this requirement, it implements the outdated Barnes-Hut approximation scheme which is unsuitable for large data sets. Other C++ implementations such as MulticoreTSNE and FIt-SNE are more suitable to 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 is especially problematic with Windows users, where the operating system does not include a C++ compiler by default and correctly configuring the command line can be an arduous process.

openTSNE is a Python package which focuses on accessibility. We provide precompiled binaries for all major Python version 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 through the conda-forge channel. openTSNE is designed to be familiar to Python users and follows the scikit-learn API [16], 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 overhead, the runtime is comparable to that of its pure 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, and has already been used by various researchers in their own work.

Comparison to UMAP

Uniform Manifold Approximation and Projection (UMAP) [2] has often been touted as a silver bullet to the problems listed above, exhibiting impressive

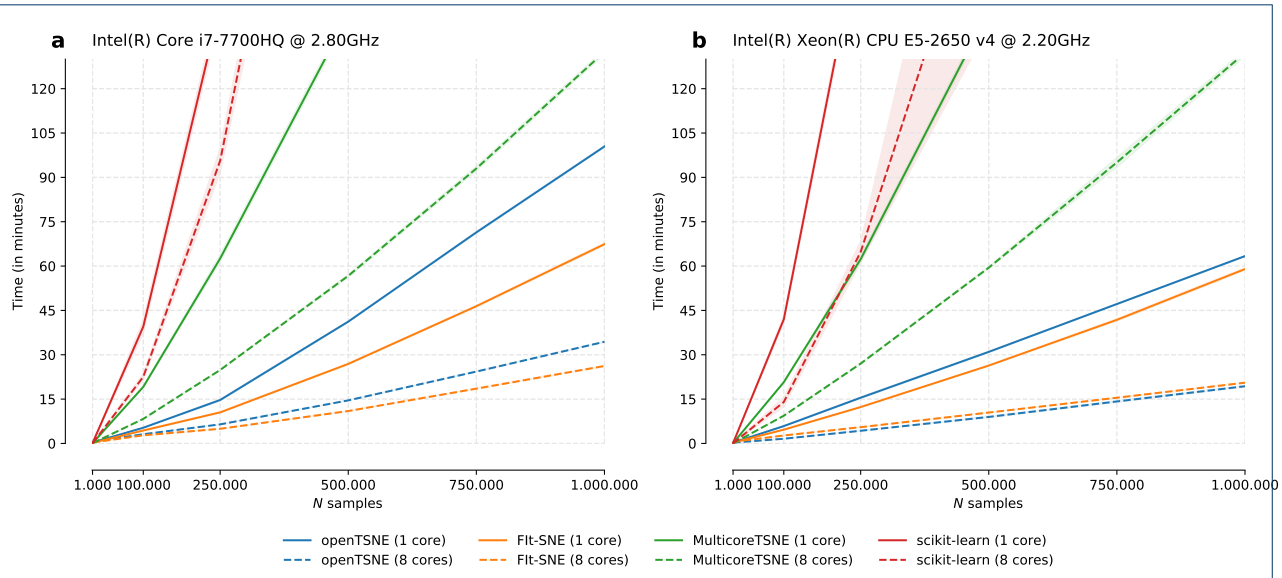

Figure 5 We benchmark openTSNE against three popular open-source implementations including scikit-learn [20], MulticoreTSNE [19], and FIt-SNE [9]. Benchmarks were run on a consumer-grade Intel Core i7-7700HQ processor found in laptop computers, and on a server-grade Intel Xeon E5-2650. To generate benchmark 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.

runtime on large data sets and organizing resulting clusters in a coherent manner. However, most performance comparisons to t-SNE were performed against older t-SNE implementations. Furthermore, the optimization procedure of UMAP is inherently sequential and does not easily lend itself to parallelization. On the other hand, modern t-SNE approximations are highly parallelizable and easily lend themselves to utilizing multiple cores, present on almost every consumer-grade processor. The embeddings produced by UMAP, like other force-directed layout algorithms such as t-SNE and ForceAtlas2 [21], are largely dependent on their initialization. Many implementations of t-SNE begin with a randomly initialized embedding, leading to a largely scattered and incoherent layout of resulting clusters. On the other hand, UMAP embeddings are initialized using Laplacian eigenmaps, a method for the preservation of global relationships between data points. Kobak & Linderman showed in [13] that the supposed superior preservation of global structure is largely the result of this initialization, and that t-SNE, if initialized in similar manner, performed just as well in preserving global coherence. Most modern implementation of t-SNE default to such initialization schemes, typically though principal component analysis or spectral embeddings.

We argue here not that either method is superior to the other, but rather, that they are complementary and should be used in tandem. Each may reveal different aspects of the data missed by the other.

Conclusion

Text for this section ...

Methods

t-SNE

Local, non-linear dimensionality reduction by t-SNE is performed as follows. Given a multi-dimensional data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^D$ where N is the number of data points in the reference data set, t-SNE aims to find a low dimensional embedding $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\} \in \mathbb{R}^d$ where $d \ll D$, such that if points $\mathbf{x}_i$ and $\mathbf{x}_j$ are close in the multi-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 t-SNE is defined as:

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

where $\mathcal{D}$ is a 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.

The similarity between points $\mathbf{y}_i$ and $\mathbf{y}_j$ in the embedding space is defined using the t-distribution with one degree of freedom

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

The t-SNE method finds an embedding $\mathbf{Y}$ that minimizes the Kullback-Leibler (KL) divergence between $\mathbf{P}$ and $\mathbf{Q}$,

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

It is then easy to show that the corresponding gradient takes on the form

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

Optimization is performed using batch gradient descent with the delta-bar-delta update rule for increased convergence speed [22]. 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 move throughout the embedding uninhibited by repulsive forces. Then the remaining 750 iterations are run with $\rho = 1$, which reverts the attractive forces to their original values.

Belkina *et al.* later found that the number of iterations can saftely be decreased to 750 iterations by increasing the learning rate to $\eta = N/12$ instead of the default and somewhat arbitrarily set $\eta = 200$ [15]. A welcome side effect of this reduction is also faster runtime.

Efficient Implementation

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 [12]. The gradient may be rewritten as

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

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. 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 realised that, qualitatively, embeddings are visually virtually indistinguishable when using only *approximate* nearest neighbors, further reducing time complexity to $\mathcal{O}(N)$ [9].

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. 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. 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 approximations 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. We would note here that while the default choices introduce some runtime overhead on smaller data sets, this effect is minimal and impacts runtime by only a few seconds compared to the exact nearest-neighbor search Barnes-Hut approximation.

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 [11]. 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 $\mathcal{O}(M)$.

openTSNE is currently the only open-source t-SNE implementation allowing users to add new samples into existing embeddings.

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-provided 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 such 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, and less overall clusters are revealed.

Kobak *et al.* [10] suggest that replacing the Gaussian kernel with a mixture of Gaussians may provide better insight into both the local and global structure. Therefore, the affinities 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 is often used to reveal 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.* [14] 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 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 result 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

	scikit-learn	MulticoreTSNE	FIt-SNE	openTSNE
PyPI package	✓	✓		✓
conda package	✓			✓
Precompiled binaries	✓			✓
$\mathcal{O}(N \log N)$	✓	✓		✓
$\mathcal{O}(N)$			✓	✓
Variable degrees of freedom			✓	✓
Variable exaggeration			✓	✓
Multiscale Gaussian kernels			✓	✓
Custom affinity kernels				✓
Adding new samples				✓

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.

Alternative Initialization Schemes

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. 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)
7. 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)
8. 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)
9. 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)
10. Kobak, D., Berens, P.: The art of using t-sne for single-cell transcriptomics. Nature Communications **10**(1), 1–14 (2019)
11. 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
12. Van Der Maaten, L.: Accelerating t-sne using tree-based algorithms. The Journal of Machine Learning Research **15**(1), 3221–3245 (2014)
13. Kobak, D., Linderman, G.C.: Umap does not preserve global structure any better than t-sne when using the same initialization. bioRxiv (2019)
14. 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)
15. 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)
16. 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)
17. 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)
18. 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)
19. Ulyanov, D.: Multicore-TSNE. GitHub (2016)
20. 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)
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. Jacobs, R.A.: Increased rates of convergence through learning rate adaptation. Neural Networks **1**(4), 295–307 (1988)

Figure 6 Sample figure title. A short description of the figure content should go here.

Figure 7 Sample figure title. Figure legend text.

Table 1 Sample table title. This is where the description of the table should go.

	B1	B2	B3
A1	0.1	0.2	0.3
A2	...	..	.
A3	..	.	.

Figures
Tables
Additional Files
Additional file 1 — Sample additional file title
Additional file descriptions text (including details of how to view the file, if it is in a non-standard format or the file extension). This might refer to a multi-page table or a figure.

Additional file 2 — Sample additional file title
Additional file descriptions text.